UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Documento di progetto di

Basi Di Dati

Corso di Laurea in Ingegneria Informatica ed Elettronica – A.A. 2020-2021

DIPARTIMENTO DI INGEGNERIA

docente

Prof. Giuseppe Liotta

Studenti

Cetra Luca 311824

Pesci Simone 310827

Tombesi Luca 311684

Sommario

Intervista

Si vuole progettare una base di dati per supportare tutte le attività amministrative e di supporto educativo nell'ambito di un programma di sostegno allo studio dedicato a studenti particolarmente meritevoli che appartengono a famiglie con ISEE compreso nelle prime due fasce (fino a circa 22 mila euro).
Il programma ha come obiettivo di accompagnare gli studenti a conseguire la laurea presso l'Università di Firenze e ha una durata di 6 anni. Il programma (per ogni studente meglio definirlo come piano formativo individuale o PFI,programma come termine è poco corretto) si compone di:

1. erogazione di una borsa di studio di importo variabile a seconda del percorso formativo (e.g. Università o scuola superiore) e della distanza della residenza dalla sede di studio. Questa borsa di studio può essere impiegata per acquistare beni e servizi in diversi ambiti attinenti al percorso formativo e personale. Le spese sostenute con le risorse della borsa di studio vanno rendicontate puntualmente per permanere nel programma.
2. Attività di mentoring finalizzati a sostenere e consigliare lo studente nel percorso formativo e stimolarlo ad ampliare i propri interessi,ambizioni e orizzonti attraverso attività esclusive e personalizzate rispetto al singolo partecipante. Queste attività si articolano in attività online,svolte nelle classi virtuali dedicate al progetto e con le piattaforme di videoconferenze e attività in presenza come visite guidate ai luoghi della cultura,ricerca e produzione particolarmente significativi in proiezione futura. Le attività possono essere inoltre sia sincrone (che richiedono la partecipazione in date e orari prestabiliti) sia asincrone (svolte in orari a scelta dello studente e quindi compatibili con i suoi impegni).

L'accesso al programma è riservato a studenti che risiedono nelle province di Arezzo Firenze e Grosseto e frequentano, al momento della domanda, il terzo o quinto anno delle scuole secondarie di secondo grado.Va però considerata la possibilità che il progetto si espanda da un contesto regionale e diventi quindi nazionale.Nelle diverse annualità del programma vengono pubblicati dei bandi per la selezione dei ragazzi che entreranno nel programma. Nel primo anno ad esempio sono stati ammessi al programma 70 studenti universitari e 90 delle secondarie. Nel secondo anno sono stati selezionati 66 studenti universitari e 80 delle secondarie. La permanenza nel programma è condizionata al raggiungimento di obiettivi di merito e di partecipazione alle attività di mentoring oltre che al mantenimento di requisiti previsti per l'accesso al programma. Tra i parametri da registrare per valutare il raggiungimento degli obiettivi di merito ci sono:
Universitari

1. CFU acquisiti sostenendo gli esami universitari
2. Media dei voti

Secondarie

1. Voto medio allo scrutinio finale
2. Voto in condotta

Per valutare la partecipazione alle attività di mentoring si registra per ogni attività significativa la partecipazione degli studenti.
Ogni attività è caratterizzata da un numero di minuti equivalenti e quindi data la partecipazione di diversi studenti alle attività si ottiene per ogni studente il numero di minuti equivalenti di partecipazione alle attività di mentoring.
La gestione e il programma di mentoring vengono curati da 3 tipologie di tutor:

1. Tutor di gestione: si occupano di tutti i processi amministrativi e di rendicontazione previsti dalle attività del programma; forniscono periodicamente agli enti finanziatori relazioni e statistiche sull'andamento delle attività e sui risultati degli studenti. Collaborano alle attività di pubblicizzazione del bando annuale e di comunicazione.
2. Tutor di merito: costituiscono il punto di riferimento per i ragazzi delle secondarie relativamente a tutti gli aspetti che condizionano il rendimento scolastico e scambiano informazioni con gli altri tutor per coordinare o per personalizzare le attività del programma. Ad ogni ragazzo è assegnato uno dei tutor.
3. Tutor online: sono responsabili della gestione delle attività di mentoring online e della proposta di stimoli e consigli per favorire la crescita personale e l'ampliamento degli orizzonti culturali

Uno stesso tutor può far parte di diversi gruppi di tutor.

Analisi delle specifiche

Frasi di carattere generale:
Si vuole progettare una base di dati per supportare tutte le attività amministrative e di supporto educativo nell'ambito di un programma di sostegno nello studio dedicato a studenti particolarmente meritevoli.

Frasi relative agli studenti: Si vuole progettare una base di dati per supportare tutte le attività amministrative e di supporto educativo nell'ambito di un programma di sostegno nello studio dedicato a studenti particolarmente meritevoli che appartengono a famiglie con ISEE compreso nelle prime due fasce (fino a 22 mila euro). Il programma ha come obiettivo di accompagnare gli studenti a conseguire la laurea presso l'Università di Firenze e ha una durata di 6 anni. L'accesso al programma è riservato a studenti che risiedono nelle province di Arezzo Firenze e Grosseto e frequentano al momento della domanda il terzo o quinto anno delle scuole secondarie di secondo grado. Nel primo anno ad esempio sono entrati 70 studenti universitari e 90 delle secondarie. Nel secondo anno sono entrati 66 studenti universitari e 80 delle secondarie. La permanenza del programma è condizionata al raggiungimento di obiettivi di merito e di partecipazione alle attività di mentoring oltre che al mantenimento di requisiti previsti per l'accesso al programma.
Per valutare la partecipazione alle attività di mentoring si registra per ogni attività significativa la partecipazione degli studenti.

Frasi relative al programma: Il programma ha come obiettivo di accompagnare gli studenti a conseguire la laurea presso l'Università di Firenze e ha una durata di 6 anni. il programma si compone di borse di studio e di attività di mentoring. L'accesso al programma è riservato a studenti che risiedono nelle province di Arezzo Firenze e Grosseto e frequentano al momento della domanda il terzo o quinto anno delle scuole secondarie di secondo grado. La permanenza del programma è condizionata al raggiungimento di obiettivi di merito e di partecipazione alle attività di mentoring oltre che al mantenimento di requisiti previsti per l'accesso al programma.

Frasi relative a borsa di studio: La borsa di studio è di importo variabile a seconda del percorso formativo (e.g. Università o scuola superiore) e della distanza della residenza dalla sede di studio. Questa borsa di studio può essere impiegata per acquistare beni e servizi in diversi ambiti attinenti al percorso formativo e personale.

Frasi relative alle attività di mentoring: Le attività di mentoring sono finalizzate a sostenere e consigliare lo studente nel percorso formativo e stimolarlo ad ampliare i propri interessi,ambizioni e orizzonti attraverso attività esclusive e personalizzate rispetto al singolo partecipante. Queste attività si articolano in attività online e attività in presenza. Le attività possono essere inoltre sia sincrone (che richiedono la partecipazione in date e orari prestabiliti) sia asincrone (svolte in orari a scelta dello studente e quindi compatibili con i suoi impegni).

Frasi relative agli attività online: Le attività online sono svolte nelle classi virtuali dedicate al progetto e con le piattaforme di videoconferenze e attività in presenza come visite guidate ai luoghi della cultura,ricerca e produzione particolarmente significativi in proiezione futura.

Frasi relative alle attività in presenza: Queste attività si articolano in attività online,svolte nelle classi virtuali dedicate al progetto e con le piattaforme di videoconferenze e attività in presenza come visite guidate ai luoghi della cultura,ricerca e produzione particolarmente significativi in proiezione futura.
Frasi relative alle annualità del programma: Vengono pubblicati dei bandi per la selezione dei ragazzi che entreranno nel programma. Ogni annualità del programma tiene traccia del numero di studenti presenti e ogni studente può partecipare a più annualità diverse.

Frasi relative agli studenti universitari: Tra i parametri da registrare per valutare il raggiungimento degli obiettivi di merito ci sono: CFU acquisiti sostenendo gli esami universitari e media dei voti

Frasi relative agli studenti delle secondarie: L'accesso al programma è riservato a studenti che risiedono nelle province di Arezzo Firenze e Grosseto e frequentano al momento della domanda il terzo o quinto anno delle scuole secondarie di secondo grado. Il raggiungimento degli obiettivi di merito da parte degli studenti delle secondarie dipendono dal voto medio allo scrutinio finale e dal voto in condotta.

Frasi relative agli obiettivi di merito: Gli obiettivi di merito variano in base a se lo studente è uno studente universitario o uno studente di una scuola secondaria.

Frasi relative alla partecipazione: La permanenza nel programma è condizionata al raggiungimento di obiettivi di merito e di partecipazione alle attività di mentoring oltre che al mantenimento di requisiti previsti per l'accesso al programma. Per valutare la partecipazione alle attività di mentoring si registra per ogni attività significativa la partecipazione degli studenti. Ogni attività è caratterizzata da un numero di minuti equivalenti e quindi data la partecipazione di diversi studenti alle attività si ottiene per ogni studente il numero di minuti equivalenti di partecipazione alle attività di mentoring.

Frasi relative ai tutor: La gestione e il programma di mentoring vengono curati da 3 tipologie di tutor. Uno stesso tutor può far parte di diversi gruppi di tutor.

> frasi relative ai tutor di gestione: si occupano di tutti i processi amministrativi e di rendicontazione previsti dalle attività del programma; forniscono periodicamente agli enti finanziatori relazioni e statistiche sull'andamento delle attività e sui risultati degli studenti. Collaborano alle attività di pubblicizzazione del bando annuale e di comunicazione.
>
> frasi relative ai tutor di merito: costituiscono il punto di riferimento per i ragazzi delle secondarie relativamente tutti gli aspetti che condizionano il rendimento scolastico e scambiano informazioni con gli altri tutor per coordinare o per personalizzare le attività del PFI.
>
> frasi relative ai tutor online: sono responsabili della gestione delle attività di mentoring online e della proposta di stimoli e consigli per favorire la crescita personale e l'ampliamento degli orizzonti culturali.

Frasi relative a regione:
Va però considerata la possibilità che il progetto si espanda da un contesto regionale e diventi quindi nazionale.

Glossario dei termini principali

Il glossario dei termini agevola la comprensione e la decomposizione delle specifiche nonché l'individuazione dei concetti più rilevanti nell'ambito di interesse.

Termine	Descrizione	Sinonimo	Collegamenti
Studente	Studente universitario o della scuola secondaria che ha accesso al Programma.	partecipante,ragazzo	PFI, annualità, obiettivi di merito, partecipazione, tutor.

PFI	Eroga borse di studio e propone attività di mentoring agli studenti.		Studente, Borsa di studio, Attività, annualità
Borsa di studio	Consente l'acquisto di beni e servizi per studenti che partecipano al programma		PFI
Attività	Sviluppano le competenze del ragazzo durante il percorso formativo.		PFI
Annualità	Edizione del programma di un dato anno.		PFI, Studente
Obiettivi di merito	Il raggiungimento condiziona la permanenza degli studenti nel programma.		Studente
Partecipazione	Partecipazione dello studente ad un'attività.		Studente, Attività
Tutor	Persona che deve guidare e consigliare lo studente.		Studente, PFI

Elenco delle operazioni

OP1: Aggiunta di uno studente (153 volte all'anno);

OP2: Aggiunta di un PFI (292 volte all'anno);

OP3: Aggiunta di un'attività all'Elenco Attività (24 volte all'anno);

OP4: Aggiunta di un'Annualità Secondaria (169 volte all'anno);

OP5: Aggiunta di un'Annualità Universitaria (123 volte all'anno);

OP6: Aggiunta di un esame superato da uno studente (5*123 = 615 volte all'anno);

OP7: Aggiunta di un Tutor (10 volte all'anno);

OP8: Modifica della residenza di uno studente (10 volte all'anno);

OP9: Aggiornamento di un PFI (584 volte all'anno);

OP10: Eliminazione di un esame superato da uno studente (123 volte all'anno);

OP11: Stampa gli studenti universitari che rispettano gli obiettivi di merito in un dato anno (12 volte all' anno);

OP12: Stampa i pfi che rispettano i minuti di partecipazione in un dato anno (12 volte all' anno);

OP13: Stampa il valore medio della media dei voti degli studenti delle scuole secondarie in un dato anno (1 volta all'anno);

OP14: Stampa gli studenti universitari che hanno superato da un numero minimo di esami ad un numero massimo di esami in un dato anno (1300 volte all'anno);

OP15: Stampa i Tutor che appartengono a più categorie di Tutor (2 volte all'anno);

OP16: Stampa i Tutor che appartengono a una sola categoria (2 volte all'anno);

OP17: Stampa il numero di partecipanti attivi della scuola secondaria in un dato anno (2 volte all'anno);

OP18: Stampa il numero di partecipanti attivi universitari in un dato anno (2 volte all'anno);

OP19: Per ogni scuola stampa quanti studenti hanno partecipato al programma in un dato anno (2 volte all'anno);

OP20: Stampa il numero di attività per ogni studente in un dato anno (12 volte all'anno);

OP21: Stampa il numero di CFU ottenuto da ogni studente universitario in un dato anno (12 volte all'anno);

OP22: Modifica lo stato di attività di uno studente universitario (4 volte all'anno);

OP23: Modifica dei CFU ottenuti da uno studente universitario in un dato esame (20 volte all'anno);

OP24: Modifica lo stato di attività di uno studente secondario (6 volte all'anno);

OP25: stampa quanti PFI ha approvato ogni Tutor di Gestione (2 volte all'anno);

OP26:Stampa il numero di studenti in un dato anno che hanno un residuo inferiore a 100€ nella borsa di studio (2 volte all'anno);

OP27:Stampa media universitario per ogni studente in un dato anno (12 volte all'anno);

OP28:Stampa per ogni categoria quanti soldi sono stati spesi complessivamente (12 volte all' anno);

OP29: Aggiunta di una spesa (1460 volte all'anno);

OP30: Aggiunta di un' attività svolta (4964 volte all'anno);

OP31: Aggiunta della supervisione da parte di un tutor di un'attività (27 volte all'anno);

OP32:Modifica della media e del voto finale di un'annualità secondaria (169 volte all'anno);

Schema Entità-Relazione

La progettazione concettuale di una base di dati consiste nella costruzione di uno schema Entità–Relazione in grado di descrivere al meglio le specifiche raccolte durante la fase preliminare. La complessità di questa operazione richiede passi successivi di raffinamento partendo dalla rappresentazione dei concetti primari.

I concetti cardine dello schema sono: **Studente**, **PFI** e **Annualita Studente**, poiché la base di dati deve descrivere il programma nei vari anni. Annualità Studente è il fulcro della base di dati il quale specifica per ogni anno quali studenti hanno preso parte al programma.

- **Studente**: identifica ciascun partecipante al programma.
- **Annualita Studente**: identifica chi partecipa al programma in un determinato anno.
- **PFI**: identifica il percorso personalizzato per ogni singolo studente in un determinato anno.

Le occorrenze della relazione “Partecipa” indicano che lo studente partecipa in un determinato anno al programma e gli è stato assegnato un PFI , il che vuol dire che soddisfa i requisiti per partecipare. Le occorrenze di “Assegnato” indicano che ad uno studente in un determinato anno è stato assegnato un PFI.

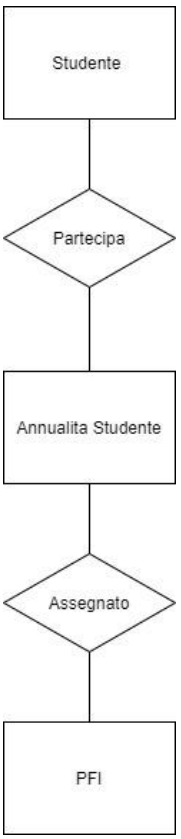

Figura 1: schema scheletro (1° livello).

Per rappresentare questa base di dati mediante uno schema E-R si è deciso di applicare una strategia di tipo “inside-out”, la quale prevede di estendere lo schema a macchia d’olio partendo da uno schema scheletro che comprende i concetti principali della base dati.
È stato deciso di dividere lo schema del livello 1 in due rami, divisi tramite la specializzazione dello studente a (Scuola secondaria/università).
Per questo motivo sono state aggiunte le seguenti entità:”Studente Scuola Secondaria”,”Studente Universitario”,”Requisiti d’accesso Universitario”,”Requisiti d’accesso Secondaria”,”Annualita Studente Secondaria” e ”Annualita Studente Universitaria”. Le seguenti entità sono state collegate mediante le seguenti relazioni:

- Anno Partecipazione Secondaria: Rappresenta l’anno a cui uno studente di scuola secondaria partecipa;
- Anno Partecipazione Universitaria: Rappresenta l’anno a cui uno studente universitario partecipa;
- Soddisfa Secondaria: Rappresenta con quali requisiti lo studente della secondaria ha avuto inizialmente accesso al programma;
- Soddisfa Universitario: Rappresenta con quali requisiti lo studente universitario ha avuto inizialmente accesso al programma ;

- Assegnazione Secondaria:Rappresenta l’assegnazione di un PFI ad uno studente secondario in un dato anno;
- Assegnazione Universitaria:Rappresenta l’assegnazione di un PFI ad uno studente universitario in un dato anno;

Le entità “Requisiti d’accesso Universitario, Requisiti d’accesso Secondaria” sono state introdotte per rappresentare le credenziali con le quali lo studente ha ottenuto l’accesso al programma.Le entità “Annualita Studente Secondaria, Annualita Studente Universitaria” sono derivate da annualità studente.

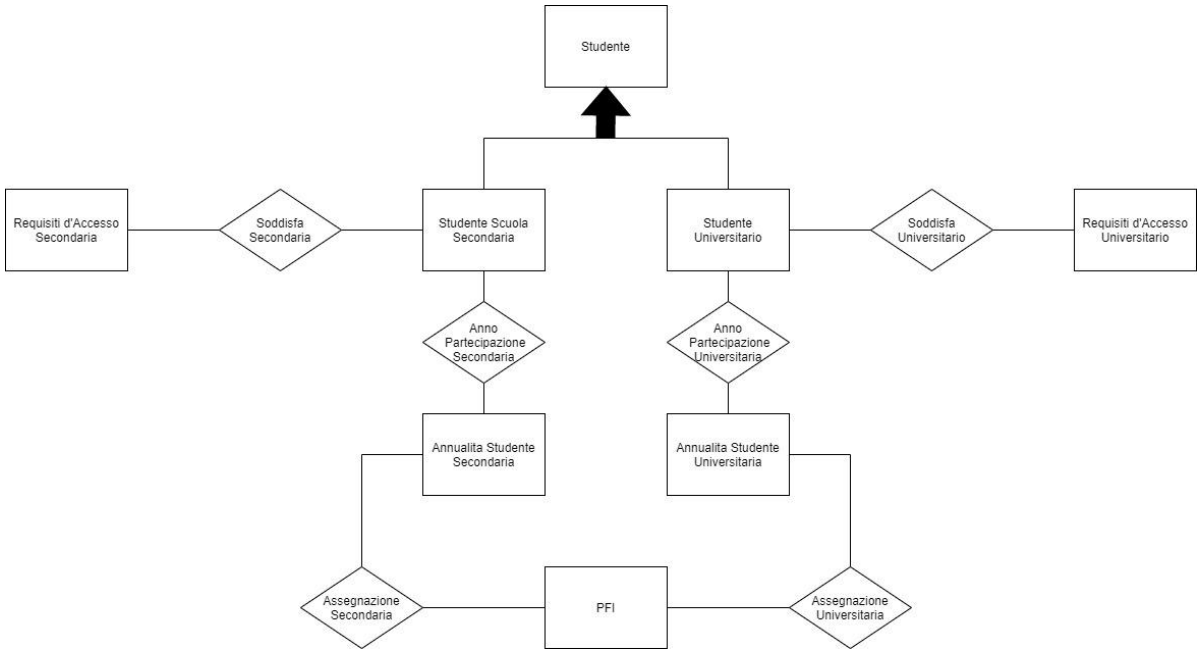

Figura 2: evoluzione schema (2° livello).

È stata aggiunta l’entità “Elenco Attivita” che rappresenta le attività che uno studente può svolgere durante l’ anno ed è stata legata a ”PFI” tramite la relazione “Attivita Svolte” . Poiché le attività possono essere online o in presenza,sono state introdotte le entità “attivita online” e “attivita in presenza” creando così una generalizzazione di “Elenco Attivita”.
Si è immesso “Comune” attraverso la relazione “Residenza” a “Studente” per sapere a quale comune appartiene lo studente. Inoltre per tenere traccia dell'indirizzo e della tipologia di scuola secondaria che lo studente frequenta è stata aggiunta l’entità “Modello Formativo”,la quale è stata legata con la relazione “Frequenta” all’entità “Studente Scuola Secondaria” e all’entità “Scuola” tramite la relazione “Offerto Da” per specificare quale scuola offrisse il modello formativo frequentato dallo studente.
Infine è stata inserita l’ entità “Provincia” con la relazione “Luogo” che lega a “Comune” specificando a quale comune appartiene la provincia ed è stata aggiunta l’entità “Regione” legata attraverso la relazione “Appartiene” a “Provincia” per indicare la regione a cui appartiene ciascuna provincia.

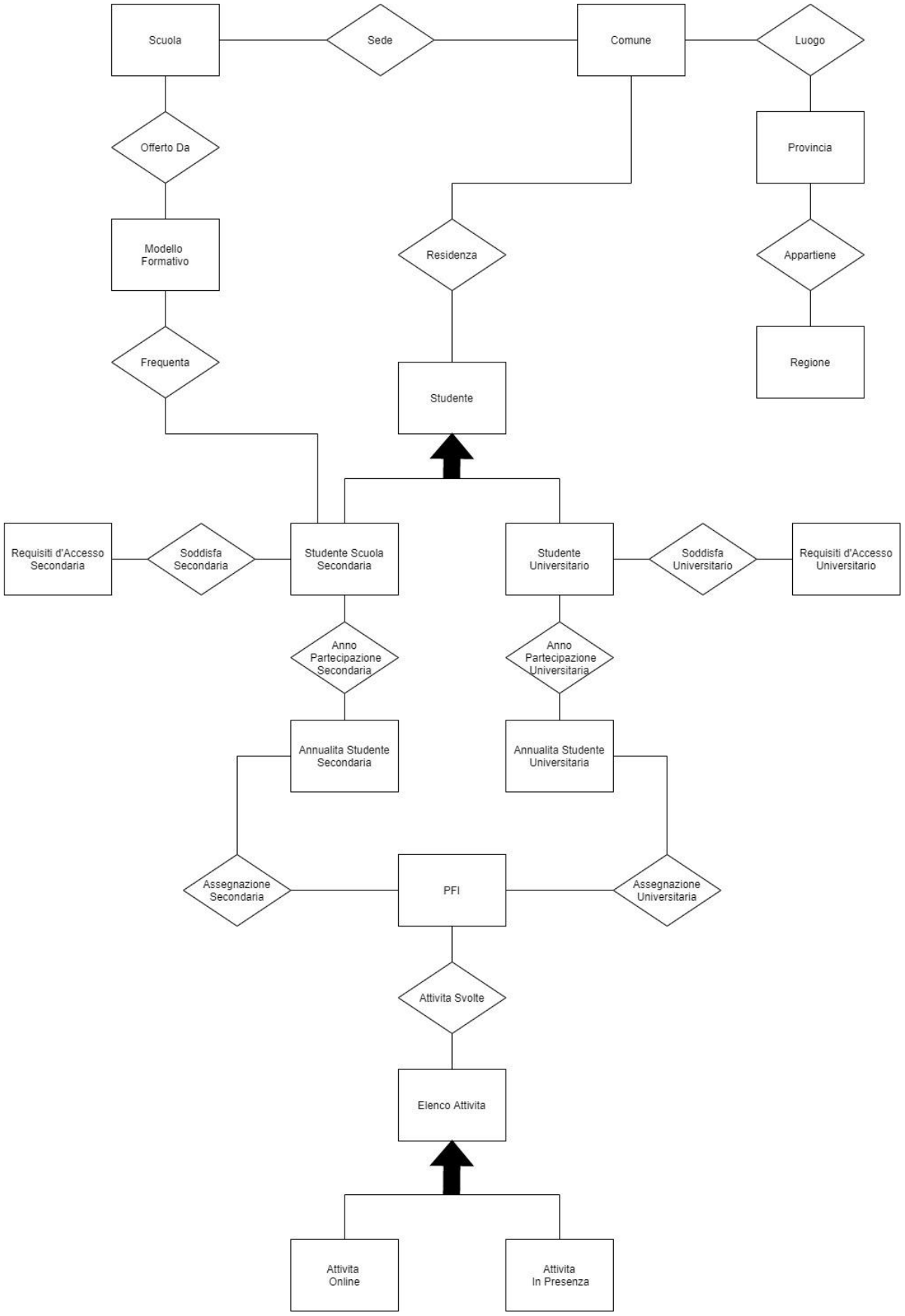

Figura 3: evoluzione schema (3° livello).

E' stata introdotta l'entità "Spese" attraverso la relazione "Acquisti" a "PFI" per monitorare le spese effettuate con la borsa di studio durante il programma.

L'entità “AreaSpese” legata a “Spese” con la relazione “Categoria” permette di specificare a quale categoria appartengono le spese dello studente.
E’ stata aggiunta l’entità “Tutor” che è una generalizzazione delle specializzazioni “Tutor Di Merito”, “Tutor Di Gestione” e “Tutor Online”;
Inoltre:

- i “Tutor Online” garantiscono che lo studente abbia svolto l’attività online;
- i “Tutor Di Gestione” approvano i PFI tramite la relazione “Approva” e presenziano le attività in presenza;
- I “Tutor Di Merito” sono assegnati solo agli studenti secondari e per questo sono stati collegati con “Annualità Secondaria” attraverso la relazione “Affiliato”;

Dal momento che si ha la necessità di conoscere quali esami lo studente universitario ha superato durante l’anno, è stata aggiunta l'entità “Esami Superati” che è stata legata ad “Annualità Universitaria” tramite la relazione “Superato”. Inoltre è stata aggiunta l’entità “Indirizzo”,che è stata collegata tramite la relazione “Tipologia” all’entità “Modello Formativo” per rappresentare tutti i possibili indirizzi di un modello formativo.

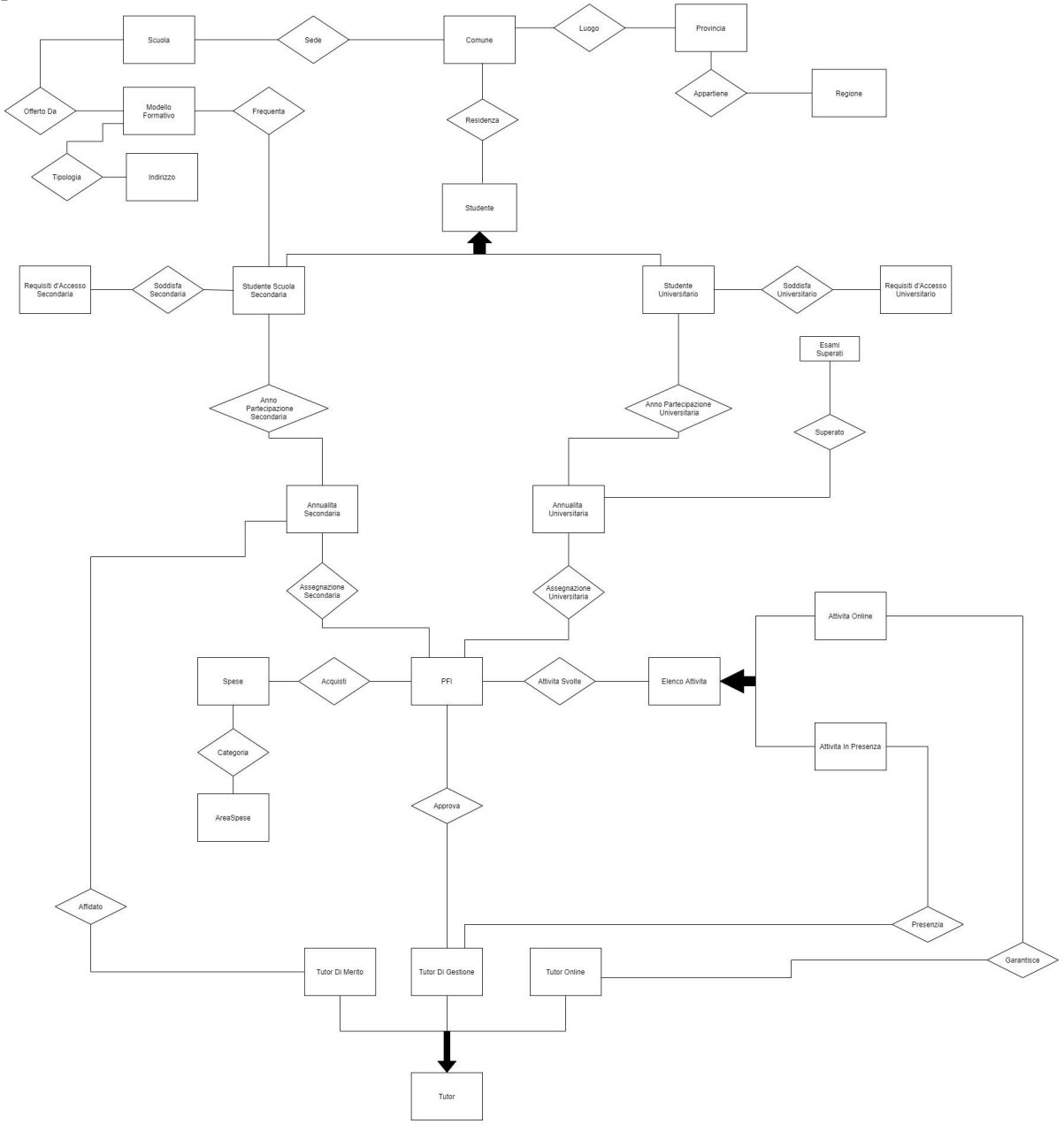

Figura 4: evoluzione schema (4° livello).

Lo schema in figura 4 è stato infine completato con l'aggiunta delle cardinalità e degli attributi, che sono stati in parte ricavati dalla raccolta dei requisiti e in parte sono stati dedotti per rappresentare in maniera più veritiera e dettagliata alcune entità.

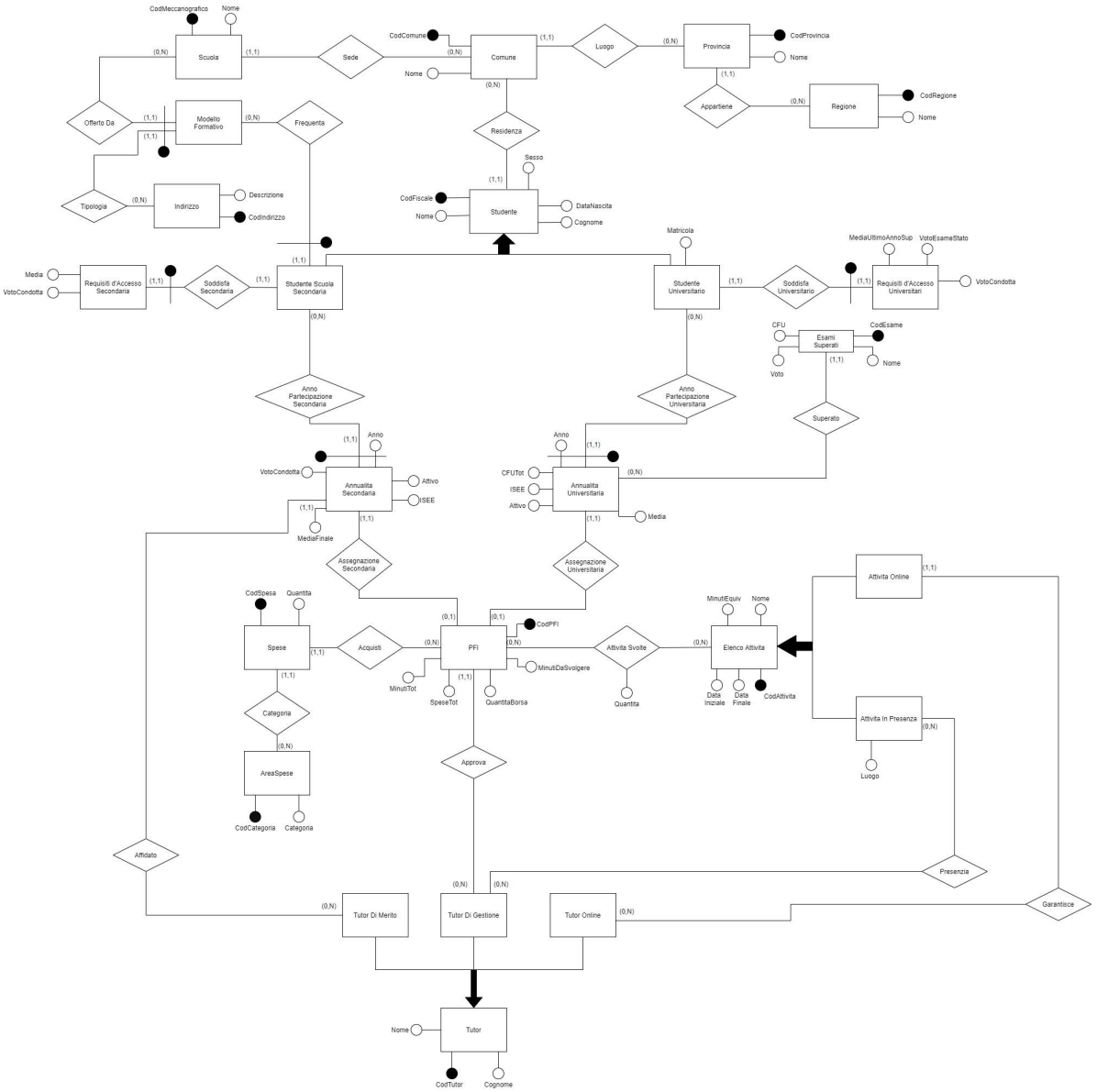

Figura 5: Schema Finale.

Vincoli non esprimibili

Non è stato possibile descrivere tramite lo schema E-R i seguenti vincoli:

- Uno studente non può partecipare al programma nella stessa annualità sia come studente universitario sia come studente della scuola secondaria.
- Ogni studente universitario deve frequentare l' università di "Firenze".
- Ogni studente ha almeno un'occorrenza di Studente Scuola Secondaria o Studente Universitario.
- Se durante l'anno gli studenti che partecipano al programma cambiano la residenza il PFI non viene alterato.
- Le attività in presenza possono essere svolte al più una volta.

Dizionario dei dati (entità)

Entità	Identificatore	Attributi	Descrizione
Provincia	CodProvincia	Nome	Provincia di un Comune
Comune	CodComune	Nome	Luogo nel quale risiede lo studente ed è situata la scuola
Regione	CodRegione	Nome	Luogo dove si trova la provincia
Studente	CodFiscale	Nome, Cognome, DataNascita, Sesso	Studente che partecipa al programma
Studente Scuola Secondaria	CodMeccanografico, Indirizzo, CodFiscale		Studente che frequenta una scuola secondaria
Modello Formativo	CodMeccanografico, Indirizzo		Tipologia ed indirizzo frequentati dallo studente in una specifica scuola
Indirizzo	CodIndirizzo	Descrizione	Identifica la tipologia di indirizzo relativa al modello formativo
Scuola	CodMeccanografico	Nome	Istituto che frequenta lo studente
Studente Universitario	CodFiscale	Matricola	Studente che frequenta l'università di Firenze
Requisiti d'Accesso Universitari	CodFiscale	VotoEsameStato,VotoCondotta, MediaUltimoAnnoSup	Requisiti con cui è stato ammesso uno studente universitario al programma
Requisiti d'Accesso Secondaria	CodMeccanografico, Indirizzo, CodFiscale	Media, VotoCondotta	Requisiti con cui è stato ammesso uno studente di una scuola secondaria al programma
Elenco Attivita	CodAttivita	Datainiziale, Datafinale, Minuti equiv, Nome	Attività cui partecipano gli studenti con relativo minutaggio
Attivita Online	CodAttivita		Attività in modalità online, può essere per uno o più partecipanti

Attivita In Presenza	CodAttivita	Luogo	Attività in presenza, per attività di gruppo
Annualita Secondaria	CodMeccanografico Indirizzo, CodFiscale, Anno	ISEE, MediaFinale, Attivo, VotoCondotta	Studente di una scuola secondaria identificato dall'anno in cui partecipa al programma
Annualita Universitaria	CodFiscale, Anno	Media, CFUTot, ISEE, Attivo	Studente dell'università di Firenze identificato dall'anno in cui partecipa al programma
Spese	CodSpesa	Quantita	Spese che lo studente sostiene grazie al PFI
Esami Superati	CodEsame	Nome, CFU, voto	Esami che lo studente universitario sostiene mentre partecipa al programma
PFI	CodPFI	MinutiTot,QuantitaBorsa, MinutiDaSvolgere, SpeseTot	Programma formativo individuale, è unico per tutti gli studenti, con relativa quantità di denaro della borsa
Tutor	CodTutor	Nome, Cognome	
Tutor Online	CodTutor		Garantisce che vengano effettuate le attività online
Tutor Di Gestione	CodTutor		Sempre presente per le attività in presenza, controlla che tutto venga gestito in modo ottimale
Tutor Di Merito	CodTutor		E' legato ad uno studente di scuola secondaria ed è unico per ogni studente (di scuola secondaria)
AreaSpese	CodCategoria	Categoria	Indica in quali aree uno studente che partecipa al programma può spendere i contributi della borsa di studio

Tab. 2: Dizionario dei dati (entità).

Dizionario dei dati (relazioni)

Relazione	Entità	Attributi	Descrizione
Luogo	Provincia, Comune		Indica la provincia relativa a ciascun Comune
Appartiene	Regione, Provincia		Indica la regione a cui appartiene ciascuna provincia
Residenza	Comune, Studente		Indica la residenza di uno studente
Soddisfa Universitario	Requisiti d'Accesso universitari, Studente Universitario		Indica i requisiti superati dagli studenti universitari che partecipano al programma
Soddisfa Secondaria	Studente Scuola Secondaria, Requisiti d'accesso Secondaria		Indica i requisiti superati dagli studenti di scuola secondaria che partecipano al programma
Anno Partecipazione Universitario	Studente Universitario, Annualita Universitaria		Identifica l'anno in cui uno studente universitario partecipa al programma
Anno Partecipazione Secondaria	Studente Scuola Secondaria, Annualita Secondaria		Identifica l'anno in cui uno studente di scuola secondaria partecipa al programma
Offerto Da	Scuola, Modello formativo		Indica quali modelli formativi offre la scuola
Tipologia	Modello Formativo, Indirizzo		Specifica la tipologia di indirizzo per ogni modello formativo
Frequenta	Modello Formativo, Studente Scuola Secondaria		Indica quale modello formativo lo studente frequenta
Sede	Scuola, Comune		Indica in quale comune è situata la scuola
Superato	Esami Superati, Annualita Universitaria		Voti degli studenti ottenuti per ogni esame dato

Presenzia	Tutor di gestione, Attivita in presenza		Monitora le attività presenziate dai tutor di gestione
Garantisce	Tutor online, Attivita online		Monitora i tutor che gestiscono le attività online
Affidato	Tutor di merito, Annualita Secondaria		Mostra per ogni studente di scuola secondaria quale tutor di merito gli è stato assegnato
Approva	PFI, Tutor di gestione		Indica quale tutor di gestione approva il PFI di ogni studente
Acquisti	PFI, Spese		Traccia le spese relative a ciascun PFI e quindi a ciascuno studente
Assegnazione Universitaria	PFI, Annualita Universitaria		Gestisce i legami tra gli studenti di scuola secondaria iscritti al programma in un certo anno e il loro PFI
Assegnazione Secondaria	PFI, Annualita Secondaria		Gestisce i legami tra gli studenti universitari iscritti al programma in un certo anno e il loro PFI
Categoria	Area Spese, Spese		Tiene traccia delle categorie nelle quali sono stati spesi i contributi della borsa di studio
Attivita svolte	Elenco Attivita,PFI	Quantita	Tiene traccia delle relazioni tra i PFI e le varie attività specificando quante volte ha fatto la relativa attivita

Tab. 3: Dizionario dei dati (relazione).

Progettazione Logica

In questa fase, al fine di creare uno schema logico consono per una traduzione diretta in schema relazionale, è necessario riorganizzare e modificare lo schema E–R ottenuto durante la progettazione concettuale.

Tavola dei Volumi

Concetti	Tipo	Volume
Provincia	E	5
Comune	E	77
Studente	E	300
Scuola	E	88
Modello Formativo	E	438
Indirizzo	E	100
Regione	E	1
Studente Scuola Secondaria	E	168
Studente Universitario	E	131
Requisiti d'Accesso Universitari	E	131
Requisiti d'Accesso Secondaria	E	168
Elenco Attivita	E	48
Attivita Online	E	40
Attivita In Presenza	E	8
Annualita Secondaria	E	258
Annualita Universitaria	E	188
Spese	E	2236
AreaSpese	E	12
Esami Superati	E	295
PFI	E	446 (PFI Attivi 292)
Tutor	E	26
Tutor Online	E	10

Tutor di Gestione	E	6
Tutor di Merito	E	10
Luogo	R	77
Residenza	R	300
Soddisfa Universita	R	131
Soddisfa Secondaria	R	168
Anno partecipazione Universitaria	R	188
Anno partecipazione Secondaria	R	258
Superato	R	5(media esami conseguiti in 2 anni per studente) * 188(Annualita Universitaria) = 944
Attivita Svolte	R	18 * 446= 8028
Presenzia	R	10
Garantisce	R	31
Affidato	R	258
Approva	R	446
Acquisti	R	5 *446(numero PFI totale) = 2195
Assegnazione Universitaria	R	188
Assegnazione Secondaria	R	258
Offerto Da	R	438
Tipologia	R	438
Frequenta	R	168
Sede	R	88
Appartiene	R	5
Categoria	R	2236

Tavola delle operazioni

Operazione	Tipo	Frequenza
OP1:Aggiunta di uno studente	I	153 volte all'anno
OP2:Aggiunta di un PFI	I	292 volte all'anno
OP3:Aggiunta di un'attività all'elenco attività	I	24 volte all'anno
OP4:Aggiunta di un'annualità secondaria	I	169 volte all'anno
OP5:Aggiunta di un'annualità Universitaria	I	123 volte all'anno
OP6:Aggiunta di un esame superato da uno studente	I	5*123 volte all'anno=615 volte all'anno
OP7:Aggiunta di un Tutor	B	10 volte all'anno
OP8:Modifica della residenza di uno studente	B	10 volte all'anno
OP9:Aggiornamento di un PFI	I	2 volte all'anno *292 (PFI attivi) = 584volte all'anno
OP10:Eliminazione di un esame superato da uno studente	I	1 volta all'anno per ogni studente universitario attivo= 123 volte all'anno
OP11:Stampa gli studenti universitari che rispettano gli obiettivi di merito in un dato anno	I	1 volta al mese = 12 volte all' anno
OP12:Stampa i pfi che rispettano i minuti di partecipazione in un dato anno	I	1 volta al mese = 12 volte all' anno
OP13:Stampa il valore medio della media dei voti degli studenti delle scuole secondarie in un dato anno	B	1 volta all'anno
OP14:Stampa gli studenti universitari che hanno superato da un numero minimo di esami ad un numero massimo di esami in un dato	I	1300 volte all'anno

anno		
OP15:Stampa i Tutor che appartengono a più categorie di Tutor	B	2 volte all'anno
OP16:Stampa i Tutor che appartengono a una sola categoria	B	2 volte all'anno
OP17:Stampa il numero di partecipanti attivi della scuola secondaria in un dato anno	B	2 volte all'anno
OP18:Stampa il numero di partecipanti attivi universitari in un dato anno	B	2 volte all'anno
OP19:Per ogni scuola stampa quanti studenti hanno partecipato al programma in un dato anno	B	2 volte all'anno
OP20:Stampa il numero di attività per ogni studente in un dato anno	I	12 volte all'anno
OP21: Stampa il numero di CFU ottenuto da ogni studente universitario in un dato anno	I	12 volte all'anno
OP22: Modifica lo stato di attività di uno studente universitario	B	4 volte all'anno
OP23:Modifica dei CFU ottenuti da uno studente universitario in un dato esame	I	20 volte all'anno
OP24: Modifica lo stato di attività di uno studente secondario	B	6 volte all'anno
OP25:stampa quanti PFI ha approvato ogni Tutor di Gestione	B	2 volte all'anno
OP26:Stampa il numero di studenti in un dato anno che hanno un residuo inferiore a 100€ nella borsa di studio	B	2 volte all'anno
OP27:Stampa media universitario per ogni studente in un dato anno	I	12 volte all'anno

OP28:Stampa per ogni categoria quanti soldi sono stati spesi complessivamente	I	1 volta al mese = 12 volte all' anno
OP29: Aggiunta di una spesa	I	292*5 volte all'anno=1460 volte all'anno
OP30:Aggiunta di un' attività svolta	I	17*(292)=4964 volte l'anno
OP31: Aggiunta della supervisione da parte di un tutor di un'attività	I	27 volte l'anno
OP32: Modifica della media e del voto finale di un'annualità secondaria	I	169 volte all'anno

Analisi delle ridondanze

Una ridondanza, in uno schema concettuale, consiste nella presenza di un dato che può essere derivato da altri dati mediante una serie di operazioni. L'analisi delle ridondanze, basata sui costi in termini di spazio e tempo, è necessaria per comprendere se una data ridondanza comporta nel complesso un vantaggio o uno svantaggio.

Una possibile ridondanza potrebbe essere quella legata alla media degli studenti universitari. Questo valore potrebbe essere dedotto andando a calcolare la media dei voti ottenuti nella relazione "Votazione" per ogni studente.

Analisi di Media in Annualita Universitaria

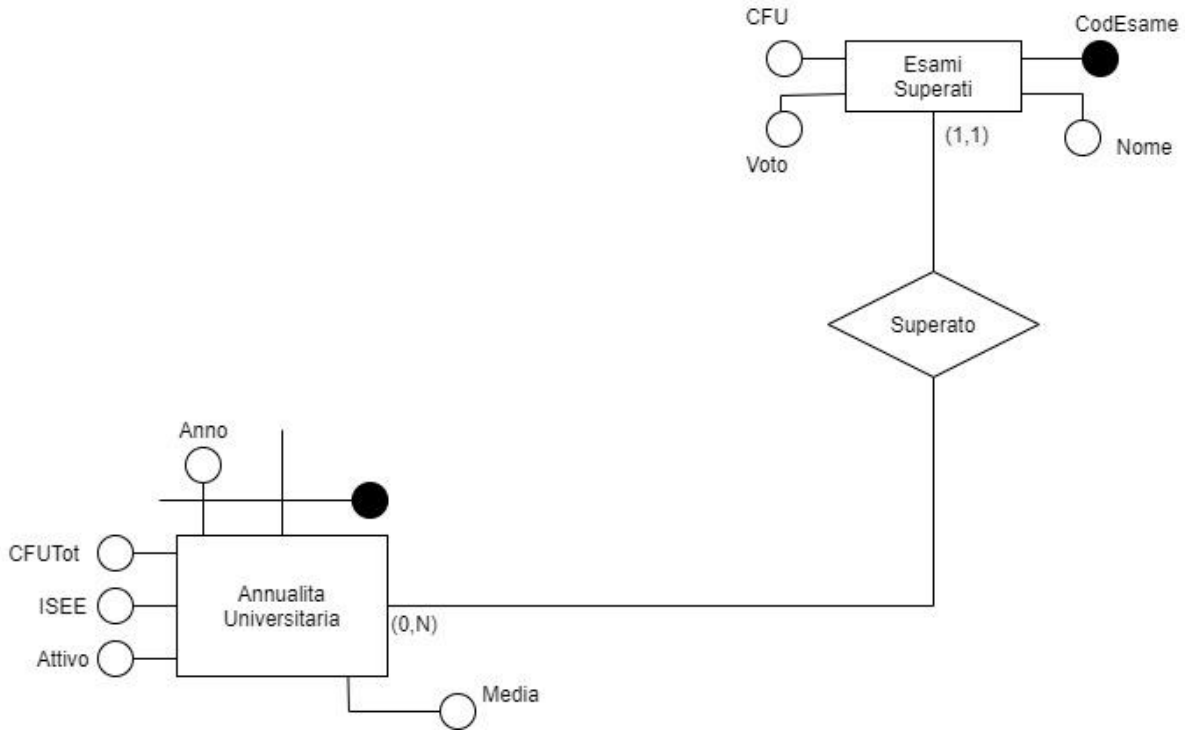

Figura 6: schema con ridondanza

Per determinare se conviene o meno mantenere questa ridondanza occorre analizzare e quindi confrontare gli indici di prestazione nei due casi, con e senza ridondanza. L'indice di prestazione viene calcolato sulla base delle operazioni che coinvolgono relazioni ed entità, oggetto della ridondanza.

In questo caso le operazioni su cui bisogna valutare gli accessi sono:

OP6: Aggiunta di un esame superato da uno studente (615 volte all'anno)
OP10: Eliminazione di un esame superato da uno studente (123 volte all'anno)
OP11: Stampa gli studenti che rispettano gli obiettivi di merito in un dato anno (12 volte all'anno)
OP26: Stampa media universitario per ogni studente (12 volte all'anno)

Si comincia analizzando gli accessi in presenza di ridondanza, cioè quando è presente l'attributo "Media" in "Annualita Universitaria".

Tavola degli accessi per OP6 (Si considera che ogni studente in un anno verbalizza 5 esami):

Concetto	**Costrutto**	**Accessi**
Esami Superati	Entità	5L+1S
Superato	Relazione	5L+1S
Annualita Universitaria	Entità	1L+1S

Tavola degli accessi per OP10 (Si considera che ogni studente in un anno verbalizza 5 esami e che in un anno si hanno 123 studenti universitari attivi):

Concetto	Costrutto	Accessi
Esami Superati	Entità	5L+1S
Superato	Relazione	5L+1S
Annualita Universitaria	Entità	1L+ 1S

Tavola degli accessi per OP11 (Si considerano 123 studenti universitari attivi all'anno):

Concetto	Costrutto	Accessi
Annualita Universitaria	Entità	1L*123=123L

Tavola degli accessi per OP26 (Si considerano 123 studenti universitari attivi all'anno):

Concetto	Costrutto	Accessi
Annualita Universitaria	Entità	1L*123=123L

Considerando il costo delle scritture doppio rispetto al costo delle letture, otteniamo il costo totale delle operazioni in presenza di ridondanza:

(11L + 3S) * 615+ (11L + 3S) * 123 +
+ (123L) * 12 +(123L)*12=15'498 accessi all'anno

Valutiamo ora i costi in assenza di ridondanza, cioè quando l'attributo "Media" non è presente.

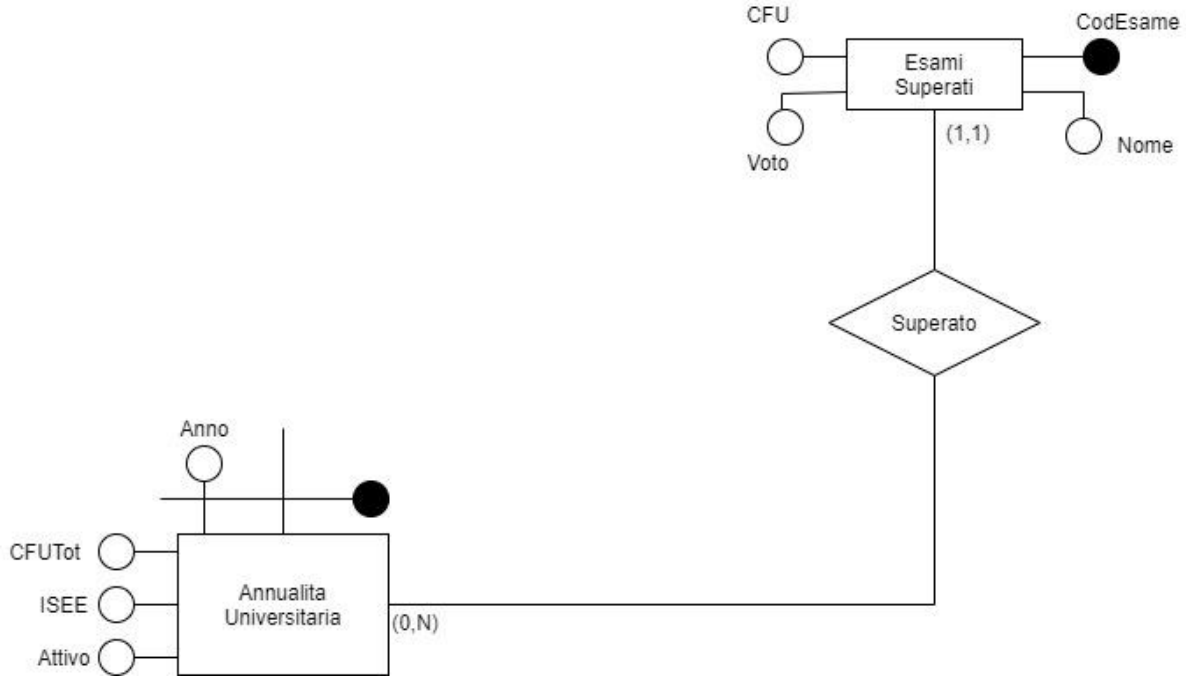

Figura 7: schema senza ridondanza

Tavola degli accessi per OP6:

Concetto	Costrutto	Accessi
Annualità Universitaria	Entità	1L
Superato	Relazione	1S
Esami Superati	Entità	1S

Tavola degli accessi per OP10:

Concetto	Costrutto	Accessi
Annualità Universitaria	Entità	1L
Superato	Relazione	1S
Esami Superati	Entità	1S

Tavola degli accessi per OP11 (Si considera che ogni studente in un anno verbalizza 5 esami e che in un anno si hanno 123 studenti universitari attivi):

Concetto	Costrutto	Accessi
Annualita Universitaria	Entità	123L
Superato	Relazione	5L*123=615L
Esami Superati	Entità	5L*123=615L

Tavola degli accessi per OP26 (Si considera che ogni studente in un anno verbalizza 5 esami e che in un anno si hanno 123 studenti universitari attivi):

Concetto	Costrutto	Accessi
Annualita Universitaria	Entità	123L
Superato	Relazione	5L*123=615L
Esami Superati	Entità	5L*123=615L

Considerando il costo delle scritture doppio rispetto al costo delle letture, otteniamo il costo totale delle operazioni in assenza di ridondanza:

(2S+1L) * 615+ (2S+1L) * 123 +
+ (615L+615L+123L) * 12 +(615L + 615L+ 123L)*12=36’162 accessi all’anno

Il mantenimento della ridondanza in annualità Universitaria provoca un costo di 1 byte*188= 188 byte in più rispetto alla variante senza ridondanza, ma poiché il numero totale di accessi è minore si è scelto di conservare la ridondanza.

Analisi di MinutiTot in PFI

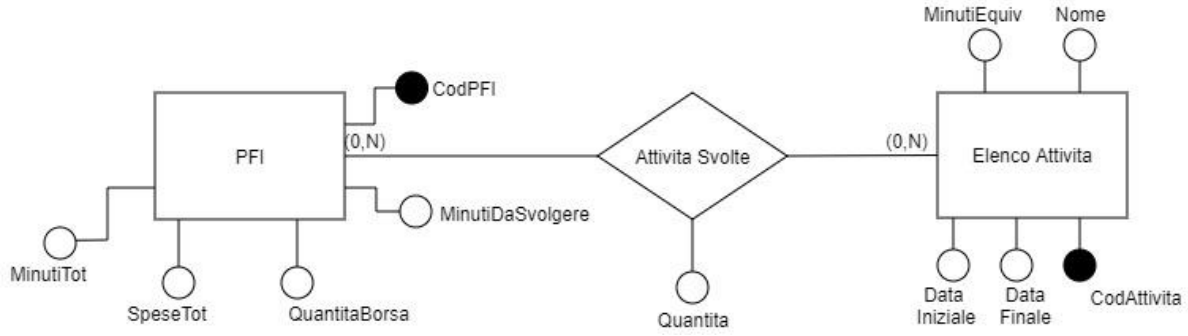

Figura 8: schema con ridondanza

Si analizza ora la presenza della ridondanza “MinutiTot” in “PFI”.

Le operazioni da analizzare sono:

OP12:Stampa gli studenti che rispettano i minuti di partecipazione in un dato anno. (12 volte all’anno)
OP29:Aggiunta attività svolta(4964 volte all’anno)

Tavola degli accessi per OP12 (292 sono i PFI attivi):

Concetto	**Costrutto**	**Accessi**
PFI	Entità	1L *(292)=292L

Tavola degli accessi per OP29:

Concetto	**Costrutto**	**Accessi**
PFI	Entità	1L+1S
Attivita svolta	Relazione	1S
Elenco Attivita	Entità	1L

Considerando il costo delle scritture doppio rispetto al costo delle letture, otteniamo il costo totale delle operazioni in presenza di ridondanza:

(1L * 292)*12 + (2L +2S)*4964 =33’288 accessi all’anno

Si analizza ora l’assenza della ridondanza “MinutiTot” in “PFI”.

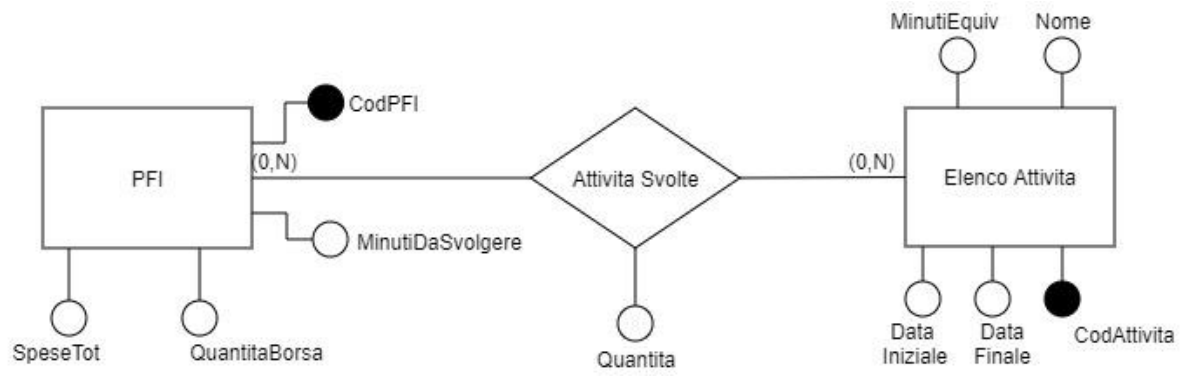

Figura 9: schema senza ridondanza

Tavola degli accessi per OP12:

Concetto	**Costrutto**	**Accessi**
PFI	Entità	1L *(292)=292L
Attivita svolta	Relazione	17L*(292)=4964L
Elenco Attivita	Entità	17L*(292)=4964L

Tavola degli accessi per OP29:

Concetto	**Costrutto**	**Accessi**
PFI	Entità	1L
Attivita svolta	Relazione	1S
Elenco Attivita	Entità	1L

Considerando il costo delle scritture doppio rispetto al costo delle letture, otteniamo il costo totale delle operazioni in assenza di ridondanza:

(292L + 4964L+4964L)*12 + (2L + 1S)*4964 = 142’496 accessi all’anno

Si nota che in presenza di ridondanza abbiamo un costo di 2 byte*446= 892byte in più per rappresentare i “MinutiTot” in “PFI”. Tuttavia il numero di accessi è sensibilmente inferiore con la presenza di ridondanza, quindi è stato scelto di mantenerla.

Analisi di CFUTot in Annualita Universitaria

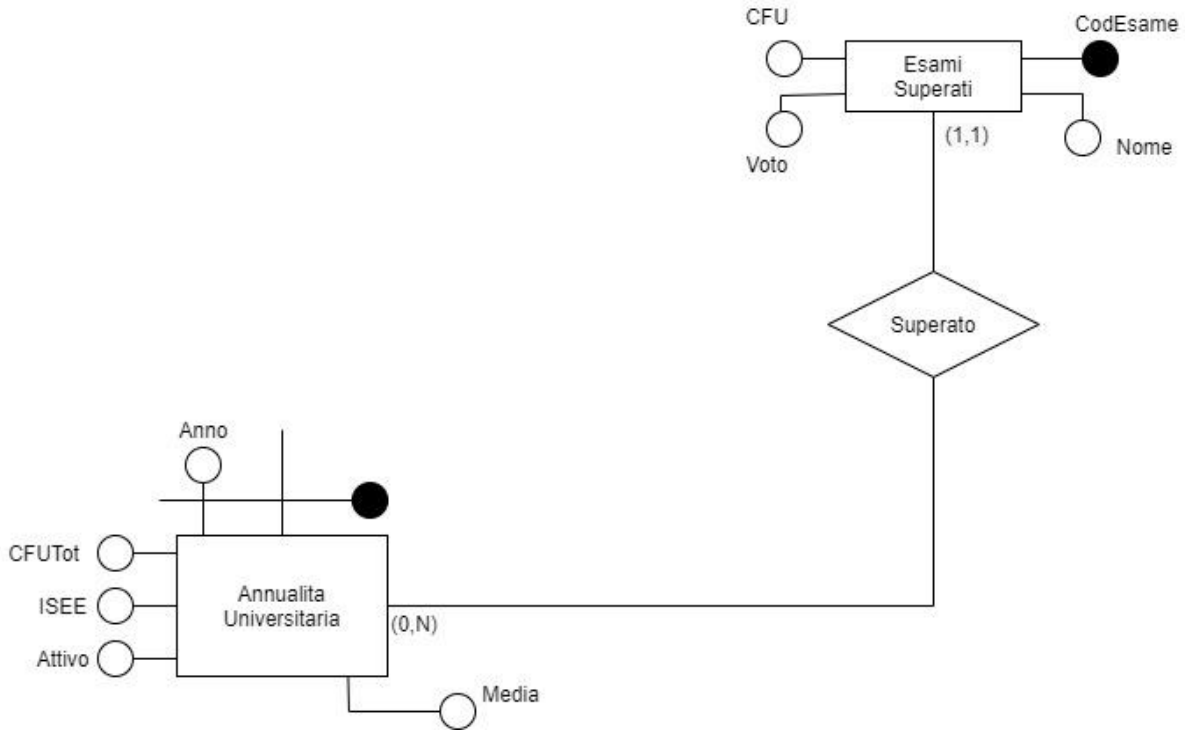

Figura 12: schema con ridondanza

Si analizza ora la presenza della ridondanza "CFUTot" in "Annualita Universitario".
Le operazioni da analizzare sono:

OP6:Aggiunta di un esame superato da uno studente (615 volte all'anno)
OP10:Eliminazione di un esame superato da uno studente (123 all'anno)
OP11:Stampa gli studenti che rispettano gli obiettivi di merito in un dato anno (12 volte all' anno)
OP21:Stampa il numero di CFU ottenuto da ogni studente universitario in un dato anno (12 volte all'anno)
OP23:Modifica dei CFU ottenuti da uno studente universitario in un dato esame (20 volte all'anno)

Si considera che ogni studente in un anno verbalizza 8 esami.

Tavola degli accessi per OP6:

Concetto	**Costrutto**	**Accessi**
Annualita Universitaria	Entità	1L+1S
Superato	Relazione	1S
Esami Superati	Entità	1S

Tavola degli accessi per OP10:

Concetto	**Costrutto**	**Accessi**
Annualita Universitaria	Entità	1L+1S

Superato	Relazione	1S
Esami Superati	Entità	1S

Tavola degli accessi per OP11 (Si considerano 123 studenti universitari attivi all'anno):

Concetto	**Costrutto**	**Accessi**
Annualita Universitaria	Entità	1L*123

Tavola degli accessi per OP21 (Si considerano 129 studenti universitari attivi all'anno):

Concetto	**Costrutto**	**Accessi**
Annualita Universitaria	Entità	1L*123

Tavola degli accessi per OP23:

Concetto	**Costrutto**	**Accessi**
Annualita Universitaria	Entità	1L+1S
Superato	Relazione	1L
Esami Superati	Entità	1L+1S

Considerando il costo delle scritture doppio rispetto al costo delle letture, otteniamo il costo totale delle operazioni in presenza di ridondanza:

(1L+3S)*615+(1L+3S)*123+(123L)*12+(123L)*12+(3L+2S)*20=8'258 accessi all'anno

Si analizza ora l'assenza della ridondanza "CFUTot" in "Annualita Universitaria".

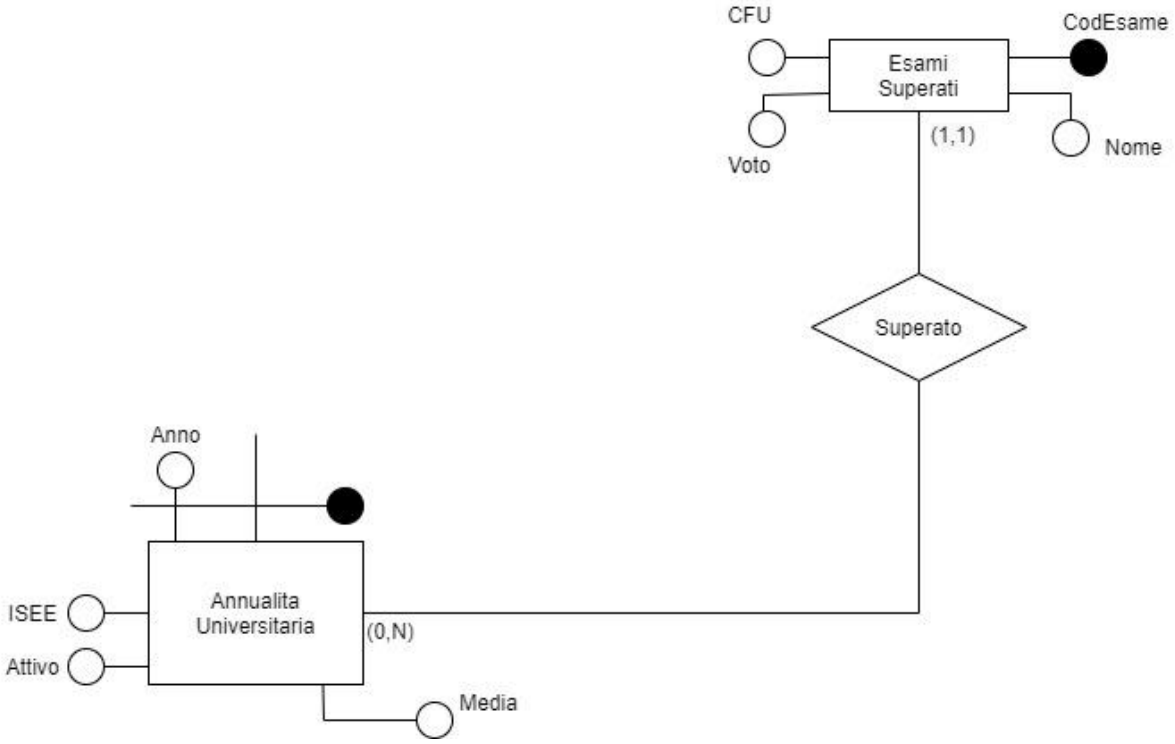

Figura 13: schema senza ridondanza

Tavola degli accessi per OP6:

Concetto	**Costrutto**	**Accessi**
Annualita Universitaria	Entità	1L
Superato	Relazione	1S
Esami Superati	Entità	1S

Tavola degli accessi per OP10:

Concetto	**Costrutto**	**Accessi**
Annualita Universitaria	Entità	1L
Superato	Relazione	1S
Esami Superati	Entità	1S

Tavola degli accessi per OP11 (Si considerano 123 studenti universitari attivi all'anno e che ognuno verbalizza in media 5 esami l'anno):

Concetto	Costrutto	Accessi
Esami Superati	Entità	5L*123=615L
Superato	Relazione	5L*123=615L
Annualita Universitaria	Entità	1L*123=123L

Tavola degli accessi per OP21 (si prendono in considerazione 5 esami superati per studente l'anno):

Concetto	Costrutto	Accessi
Annualita Universitaria	Entità	1L*123=123L
Superato	Relazione	5L*123=615L
Esami superati	Entità	5L*123=615L

Tavola degli accessi per OP23:

Concetto	Costrutto	Accessi
Esami svolti	Entità	1L+ 1S
Superato	Relazione	1L
Annualita Universitaria	Entità	1L

Considerando il costo delle scritture doppio rispetto al costo delle letture, otteniamo il costo totale delle operazioni in assenza di ridondanza:

(1L+2S)*615+(1L+2S)*123+(615L+615L+123L)*12+
+(123L+615L+615L)*12+(3L+1S)*20=36'262 accessi all'anno

Il mantenimento della ridondanza in annualità Universitaria provoca un costo di 1 byte*188= 188 byte in più rispetto alla variante senza ridondanza, ma poiché il numero totale di accessi è di molto inferiore si è scelto di conservare la ridondanza.

Analisi di SpeseTot in PFI

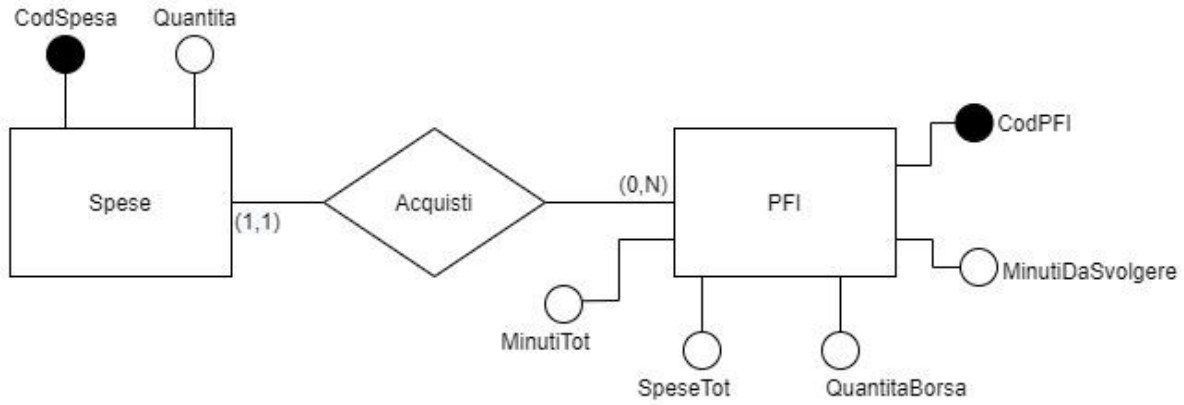

Figura 14: schema con ridondanza

Si analizza ora la presenza della ridondanza "SpeseTot" in "PFI".

Le operazioni da analizzare sono:

OP25:Stampa il numero di studenti in un dato anno che hanno un residuo inferiore a 100€ nella borsa di studio (2 volte all'anno)
OP28:Aggiunta di una spesa (1460 volte all'anno).

Tavola degli accessi per OP25 (Si considerano 292 PFI l'anno):

Concetto	**Costrutto**	**Accessi**
PFI	Entità	1L*292 = 292L

Tavola degli accessi per OP28:

Concetto	**Costrutto**	**Accessi**
Spese	Entità	1S
Acquisti	Relazione	1S
PFI	Entità	1L + 1S

Considerando il costo delle scritture doppio rispetto al costo delle letture, otteniamo il costo totale delle operazioni in presenza di ridondanza:

(292L) * 2 + (1L + 3S) * 1460 = 10'804 accessi all'anno

Si analizza ora l'assenza della ridondanza "SpeseTot" in "PFI".

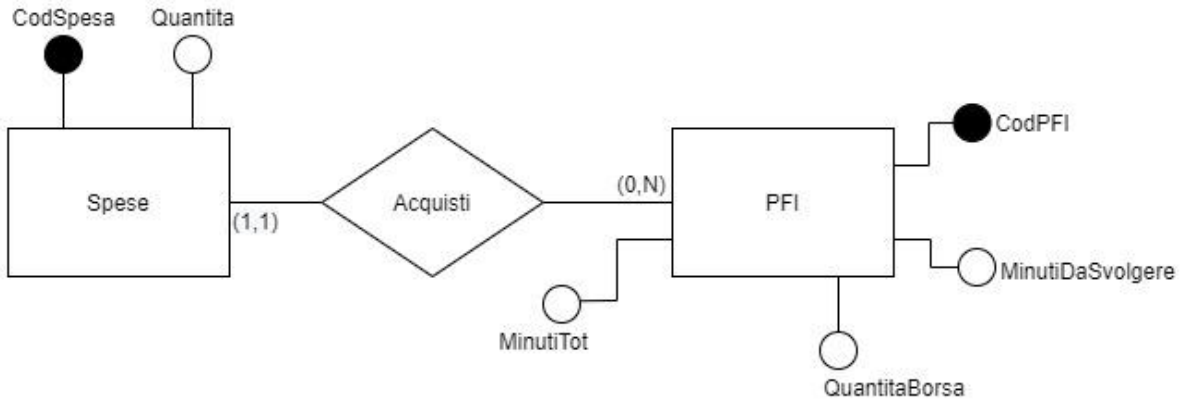

Figura 15: schema senza ridondanza

Tavola degli accessi per OP25 (si prendono in considerazione 5 spese di media per ogni PFI avendo 292 PFI l'anno):

Concetto	**Costrutto**	**Accessi**
Spese	Entità	5L*292=1460L
Acquisti	Relazione	5L*292=1460L
PFI	Entità	1L*292 = 292L

Tavola degli accessi per OP28:

Concetto	**Costrutto**	**Accessi**
Spese	Entità	1S
Acquisti	Relazione	1S
PFI	Entità	1L

Considerando il costo delle scritture doppio rispetto al costo delle letture, otteniamo il costo totale delle operazioni in assenza di ridondanza:

(1460L+1460L+292L)*2+(1L+2S)*1460=13'724 accessi all'anno

Il mantenimento della ridondanza in PFI provoca un costo di 3 byte*446= 1338 byte in più rispetto alla variante senza ridondanza, ma poiché il numero totale di accessi è inferiore, perciò si è scelto di conservare la ridondanza.

Eliminazione delle Generalizzazioni

Le generalizzazioni non sono direttamente rappresentabili nei DBMS. Per questo motivo si cerca di trasformare tali costrutti tramite l'utilizzo esclusivo di entità e relazioni.

Generalizzazione di "Tutor"

Lo schema presenta tre generalizzazioni totali:Tutor, Attivita Svolte e Studenti.

Si procede con l'analisi della generalizzazione dell'entità "Tutor".

Dato che le operazioni non fanno molta distinzione tra le occorrenze delle varie tipologie di tutor e siccome non abbiamo attributi specifici per le occorrenze delle entità figlie di Tutor. Un possibile accorpamento inoltre non porterebbe ad alcuno spreco di memoria in quanto cambierebbe solo la chiave dell'entità "Tutor"; Infatti la nuova chiave sarebbe formata dalla coppia di attributi (CodTutor,Categoria), quindi si decide di optare per un accorpamento delle entità figlie nel genitore.

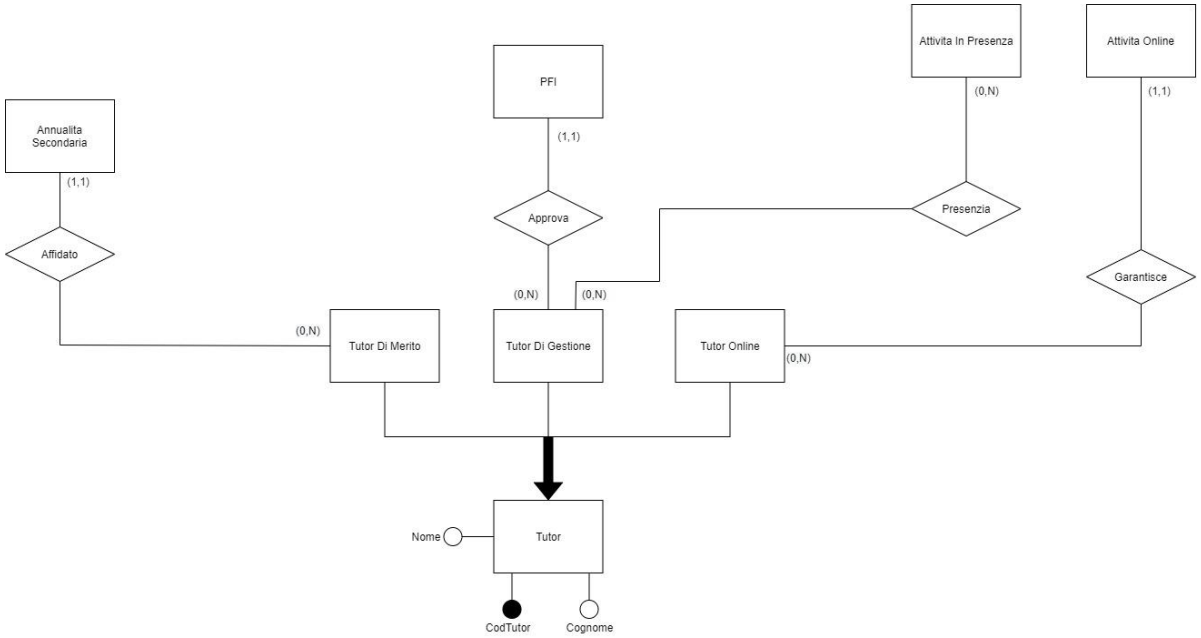

Figura 16a: eliminazione della generalizzazione "Tutor"

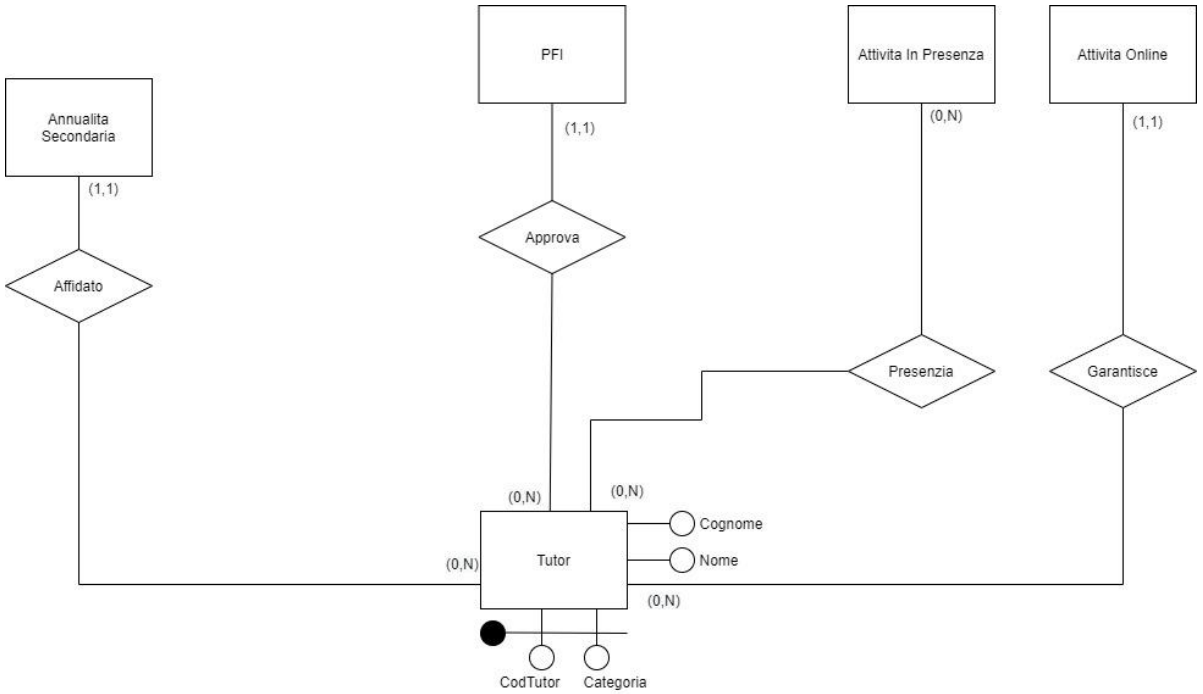

Figura 16b: accorpamento delle figlie nel genitore "Tutor"

Generalizzazione di "Elenco Attivita"

Per quanto riguarda la generalizzazione "Elenco Attivita", le operazioni non fanno distinzione tra le occorrenze delle due tipologie di attività e inoltre le due specializzazioni non hanno troppi attributi eterogenei. Inoltre le ennuple di "Attivita Online" sono in media l'80% delle ennuple di "Elenco Attivita", quindi un possibile accorpamento provocherebbe uno spreco di memoria poiché le attività online sono sprovviste di un campo luogo. Dal momento che si considera che per descrivere un luogo in cui viene svolta l'attività si utilizzi un massimo di 20 caratteri (con 16 bit per carattere secondo la codifica UNICODE), si ha uno spreco in byte di (20*16)/8 byte= 40 byte per ogni attività online e quindi provoca uno spreco totale di 40*40 byte= 1600 byte. Alla luce di queste considerazioni e poiché lo spreco di memoria è minimo si è scelto di accorpare le entità figlie all'entità padre.

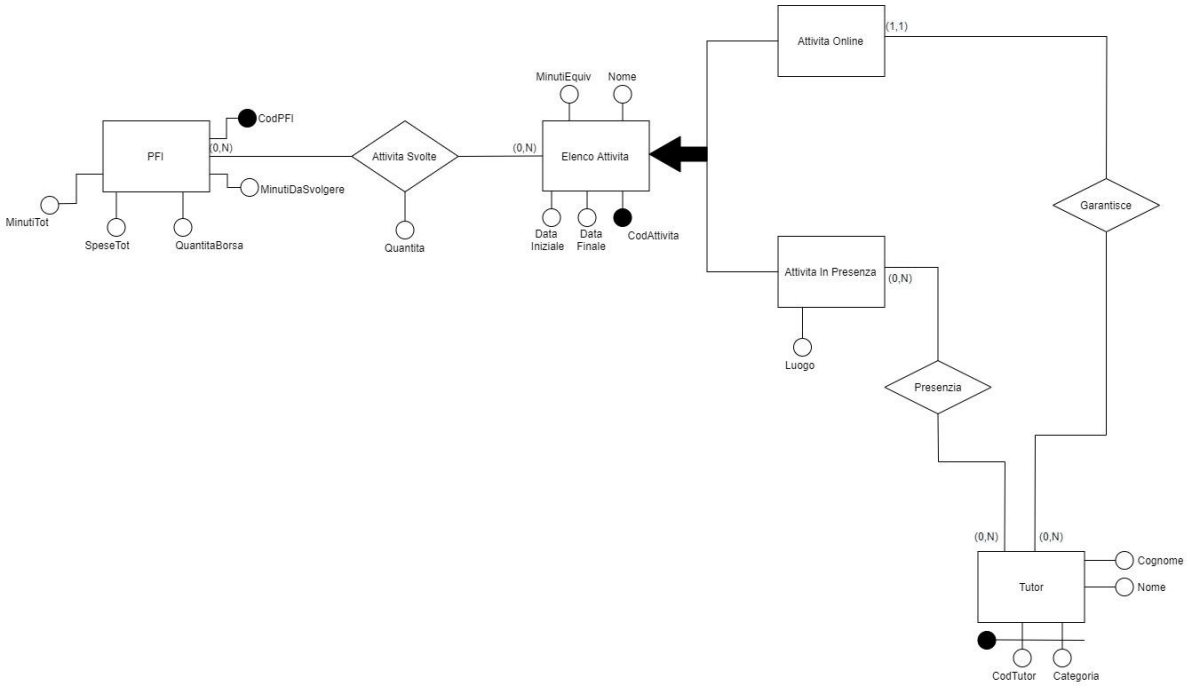

Figura 17a: eliminazione della generalizzazione "Elenco Attivita"

Per quanto riguarda le Relazioni "Presenzia" e "Garantisce" esse sono state accorpate nella relazione "Supervisione", una relazione più generalizzata che contiene dei vincoli inesprimibili quali che i tutor che vi si trovano devono essere di Categoria "G" o "O" e le attività che presenziano possono essere solo di loro competenza.

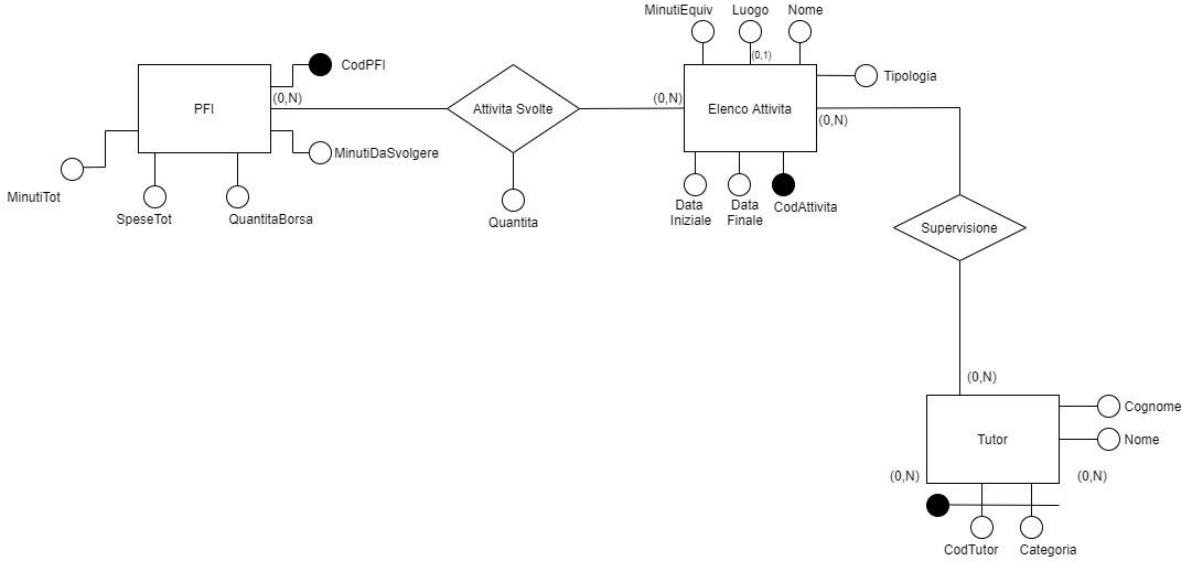

Figura 17b: accorpamento delle figlie nel genitore "Elenco Attivita"

Generalizzazione di "Studente"

Relativamente alla generalizzazione di "Studente" si è deciso di sostituire la generalizzazione con due nuove relazioni "Frequenta Secondaria" e "Frequenta Universita", poiché le due specializzazioni hanno caratteristiche molto eterogenee tra loro, infatti lo "Studente Scuola Secondaria" ha come chiave oltre al codice fiscale, il modello formativo che frequenta, invece lo studente universitario ha come attributo aggiuntivo il numero di matricola ed è proprio per questo che non si è scelto di accorpare le figlie nel genitore. Infine non si è scelto di accorpare il genitore nelle figlie poichè uno studente che partecipa al programma può appartenere sia all'entità studente scuola secondaria sia

all’entità studente universitario, in quanto uno studente per esempio dopo aver frequentato la scuola secondaria può permanere nel programma negli anni successivi come studente universitario.

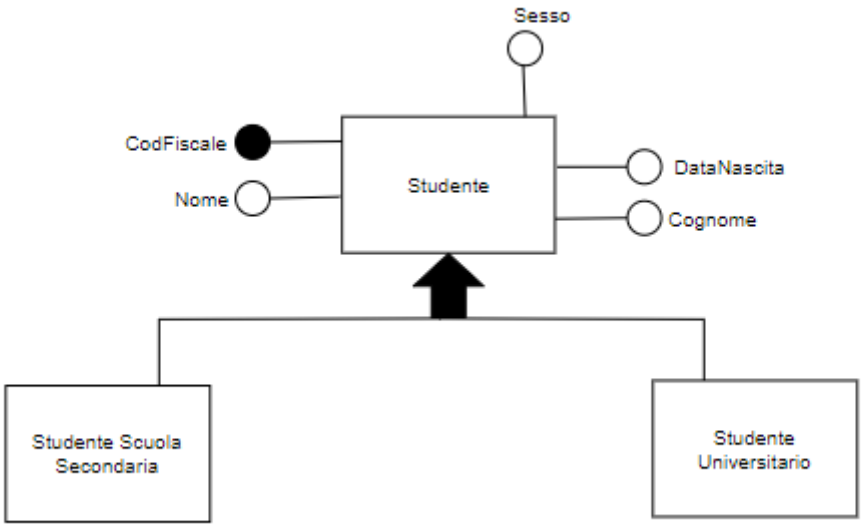

Figura 18a: schema ristrutturato per la generalizzazione “Studente”

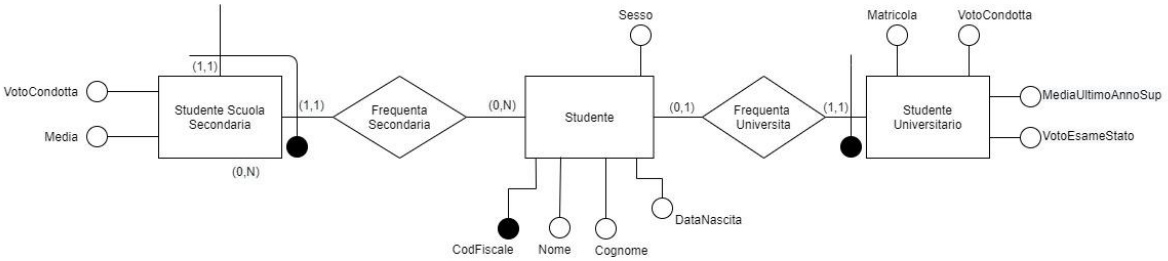

Figura 18b: schema ristrutturato per la generalizzazione “Studente”

Accorpamento/partizionamento di relationship

Il partizionamento di entità e relazioni può essere verticale, se il concetto è suddiviso sui suoi attributi, oppure orizzontale se si introduce una suddivisione basata sulle diverse occorrenze. L'accorpamento ne rappresenta l'operazione inversa. Tali procedure sono volte al raggiungimento di una maggior efficienza nell'esecuzione delle operazioni.

Andando ad analizzare lo schema E-R sono state individuate due entità che potevano essere accorpate; rispettivamente "Requisiti secondaria" a "Studenti Scuola secondaria" e "Requisiti universitari" a "Studenti universitari". Dal momento che sia "Studenti Scuola secondaria" sia "Studenti universitari" non hanno altri attributi oltre alla chiave e in entrambi i casi non ci sono operazioni specifiche che riguardano solo gli attributi di una delle due entità partizionate, si è deciso di accorpare sia "Requisiti universitari" a "Studenti universitari" sia "Requisiti secondaria" a "Studenti Scuola secondaria".

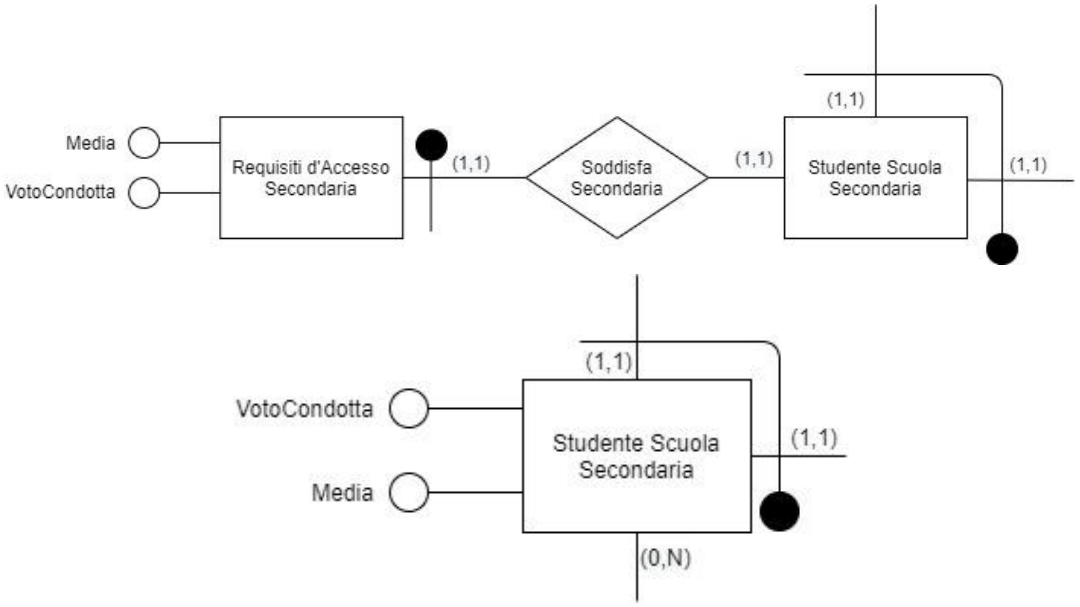

Figura 19-20: prima e dopo l'accorpamento

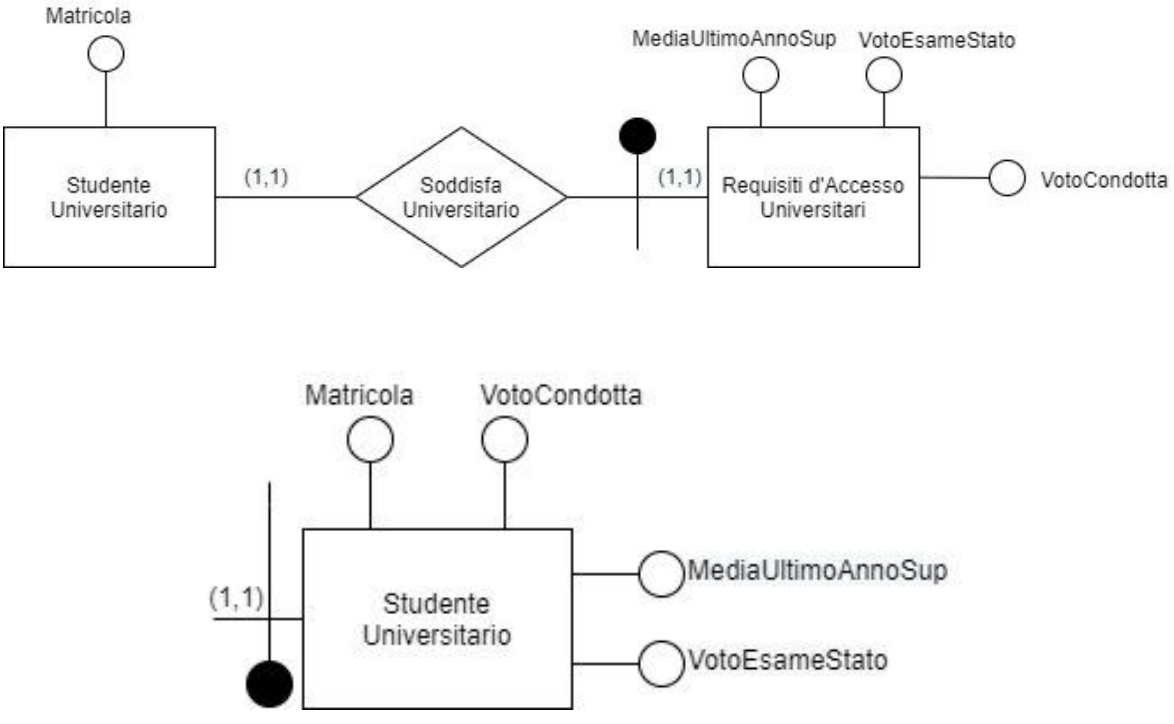

Figura 21-22: prima e dopo l'accorpamento

Scelta degli identificatori primari

La scelta degli identificatori principali è essenziale nelle traduzioni verso il modello relazionale poiché in tale modello le chiavi vengono utilizzate per stabilire legami tra dati in relazioni diverse.

Osservando lo schema di figura 23 possiamo constatare che tutte le tabelle hanno un unico identificatore interno o esterno che diventa automaticamente l'identificatore principale.

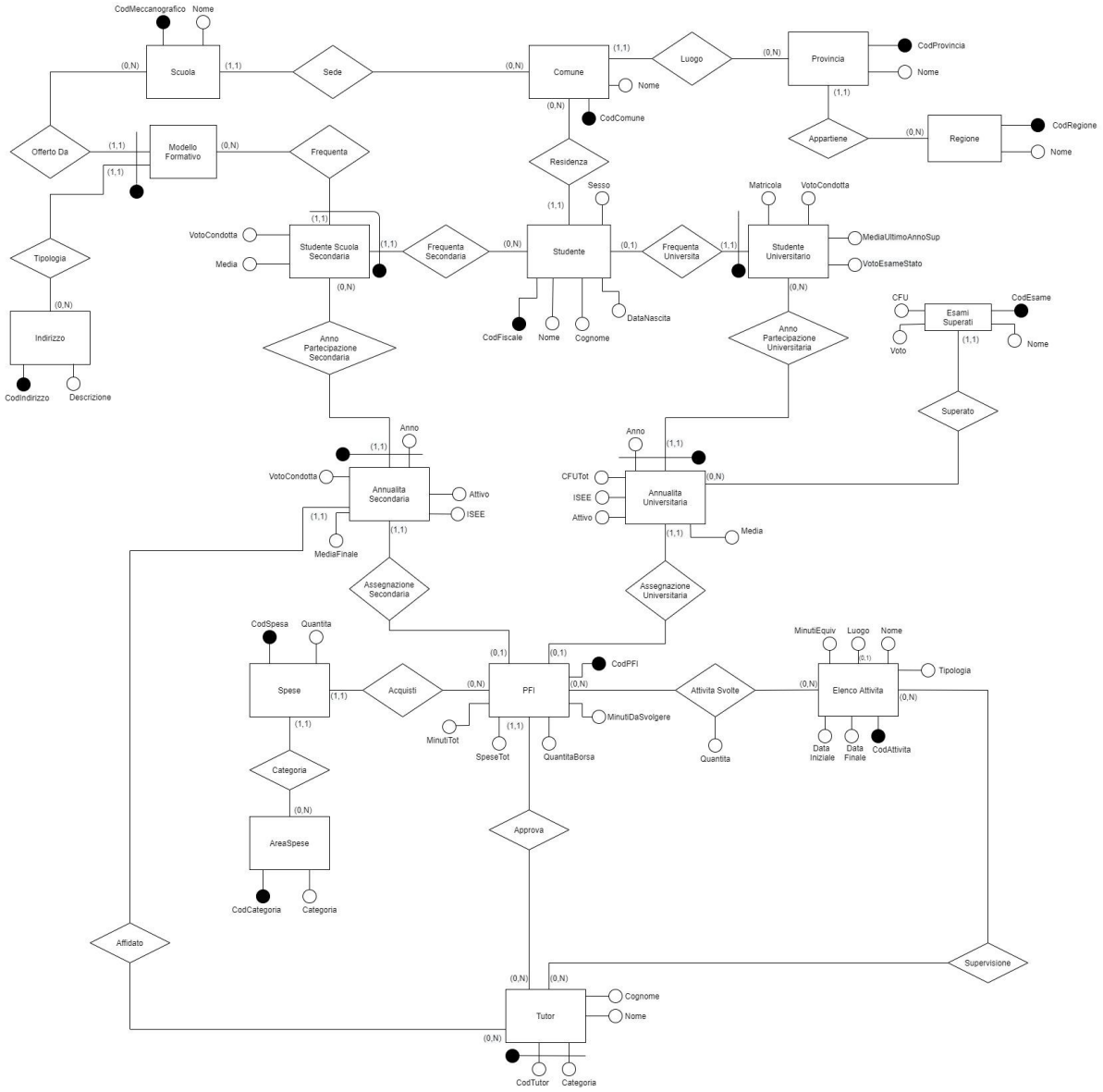

Figura 23: Schema finale ristrutturato

Modello Relazionale

A partire dallo schema E–R ristrutturato si costruisce uno schema logico equivalente in grado di rappresentare le medesime informazioni. Tale schema, chiamato modello relazione, costituisce il risultato della progettazione logica.

Traduzione di Entità:

- ELENCO ATTIVITA(CodAttivita,Tipologia,Data iniziale,Data finale,Nome,MinutiEquiv,Luogo*)

- COMUNE(CodCitta,Nome,Provincia)

  Vincolo di integrità referenziale fra l'attributo Provincia e la relazione Provincia

- PROVINCIA(CodProvincia,Nome,Regione)

  Vincolo di integrità referenziale tra l'attributo Regione e la relazione Regione

- SCUOLA(CodMeccanografico,Nome,Comune)

  Vincolo di integrità referenziale fra l'attributo Comune e la relazione Comune

- MODELLO FORMATIVO(CodMeccanografico,Indirizzo)

  Vincolo di integrità referenziale fra l'attributo CodMeccanografico e la relazione Scuola , tra l'attributo Indirizzo e la relazione Indirizzo

- REGIONE(CodRegione, Nome)

- STUDENTE(CodFiscale,Nome,Cognome,DataNascita,Sesso,Comune)

  Vincolo di integrità referenziale fra l'attributo Comune e la relazione Comune

- STUDENTE SCUOLA SECONDARIA(Studente,CodMeccanografico,Indirizzo,VotoCondotta,Media)

  Vincolo di integrità referenziale tra l'attributo Studente e la relazione Studente e tra CodMeccanografico,Indirizzo e la relazione ModelloFormativo

- INDIRIZZO(CodIndirizzo, Descrizione)

- STUDENTE UNIVERSITARIO(Studente,Matricola,VotoCondotta,MediaUltimoAnnoSup,VotoEsameStato)

  Vincolo di integrità referenziale tra l'attributo Studente e la relazione Studente

- TUTOR(CodTutor,Categoria,Cognome,Nome)

- PFI(CodPFI,MinutiDaSvolgere,QuantitaBorsa,SpeseTot,CodTutor,Categoria,MinutiTot)

  Vincolo di integrità referenziale tra l'attributo CodTutor,Categoria e la relazione Tutor

- SPESE(CodSpesa,Quantita,AreaSpese,PFI)

Vincolo di integrità referenziale tra l'attributo PFI e la relazione PFI e tra l'attributo AreaSpese e la relazione AreaSpese

- AREASPESE(CodCategoria,Categoria)

- ANNUALITA UNIVERSITARIA(Studente Universitario,Anno,ISEE,CFUTot,Media,Attivo,PFI)

  Vincolo di integrità referenziale tra l'attributo Studente Universitario e la relazione Studente Universitario e tra l'attributo PFI e la relazione PFI

- ANNUALITA SECONDARIA(Studente,CodMeccanografico,Indirizzo,Anno,ISEE,votoCondotta,Media,Attivo,codtutor,categoria,PFI)

  Vincolo di integrità referenziale tra l'attributo Studente scuola Secondario e la relazione Studente, tra l'attributo CodMeccanografico e Indirizzo e la relazione ModelloFormativo, tra l'attributo PFI e la relazione PFI
  e tra gli attributi codtutor,categoria e la relazione Tutor

- ESAMI SUPERATI(CodEsame,Nome,CFU,Voto,Studente, Anno)

  Vincolo di integrità referenziale fra l'attributo Studente, Anno e la relazione AnnualitaUniversitaria

Traduzione di Relazioni:

- SUPERVISIONE(CodTutor,Categoria,Attivita)

  Vincolo di integrità referenziale tra CodTutor, Categoria e la relazione Tutor e tra l'attributo Attivita e la relazione Elenco Attivita

- ATTIVITA SVOLTE (PFI,ElencoAttivita,Quantità)

  Vincolo di integrità referenziale tra PFI e la relazione PFI e tra l'attributo ElencoAttivita e la relazione Elenco Attivita

Normalizzazione

Per valutare la qualità della base dati si procede alla sua normalizzazione, verificando che le dipendenze funzionali includano come implicanti solo termini chiave delle relazioni (Forma Normale di Boyce e Codd).

Osservando le relazioni create in fase di traduzione si può notare come tale proprietà sia soddisfatta.

Creazione delle Tabelle

CREATE TABLE "ElencoAttivita"

```
(
      CodAttivita varchar(5) PRIMARY KEY,
      Tipologia varchar(8) NOT NULL,
      DataIniziale date NOT NULL,
      DataFinale date NOT NULL,
      Nome varchar(20) NOT NULL,
      MinutiEquiv smallint NOT NULL,
      Luogo varchar(20),
      UNIQUE(DataIniziale, DataFinale, Nome)
)
```

CREATE TABLE "Regione"

```
(
      CodRegione char(4) PRIMARY KEY,
      Nome varchar(20) NOT NULL,
      UNIQUE(Nome)
)
```

CREATE TABLE "Provincia"

```
(
      CodProvincia char(2) PRIMARY KEY,
      Nome varchar(20) NOT NULL,
      Regione char(4) NOT NULL
            REFERENCES "Regione"(CodRegione)
                  ON UPDATE CASCADE ON DELETE NO ACTION ,
      UNIQUE(Nome)

)
```

CREATE TABLE "Comune"

```
(
```

```
      CodComune varchar(6) PRIMARY KEY,
      Nome varchar(20) NOT NULL,
      Provincia char(2) NOT NULL
            REFERENCES "Provincia"(CodProvincia)
                  ON UPDATE CASCADE    ON DELETE NO ACTION,
      UNIQUE(Nome)
)
```

CREATE TABLE "Scuola"

```
(
      CodMeccanografico char(10) PRIMARY KEY,
      Nome varchar(60) NOT NULL,
      Comune varchar(6) NOT NULL
            REFERENCES "Comune"(CodComune)
                  ON UPDATE CASCADE ON DELETE NO ACTION,
      UNIQUE(Nome)

)
```

CREATE TABLE "ModelloFormativo"

```
(
      CodMeccanografico char(10),
      Indirizzo char(4),
      PRIMARY KEY(CodMeccanografico,Indirizzo),
      FOREIGN KEY(CodMeccanografico)
            REFERENCES "Scuola"(CodMeccanografico)
                  ON UPDATE CASCADE ON DELETE NO ACTION,
      FOREIGN KEY(Indirizzo)
            REFERENCES "Indirizzo" (CodIndirizzo)
                  ON UPDATE CASCADE    ON DELETE NO ACTION

)
```

CREATE TABLE "Studente"

```
(
      CodFiscale char(16) PRIMARY KEY,
      Nome varchar(30) NOT NULL,
      Cognome varchar(30) NOT NULL,
      DataNascita date NOT NULL,
      Sesso char(1) NOT NULL,
      Comune varchar(6)
            REFERENCES "Comune"(CodComune)
                  ON UPDATE CASCADE ON DELETE NO ACTION,
      UNIQUE(Nome, Cognome, DataNascita)

)
```

CREATE TABLE "StudenteScuolaSecondaria"

```
(
      CodMeccanografico char(10),
      Indirizzo char(4),
      Studente char(16)
            REFERENCES "Studente"(codFiscale)
                  ON UPDATE NO ACTION  ON DELETE NO ACTION,
      votoCondotta smallint NOT NULL,
      media numeric(2,2) NOT NULL,
      PRIMARY KEY(Studente,CodMeccanografico,Indirizzo),
      FOREIGN KEY (CodMeccanografico,Indirizzo)
            REFERENCES
      "ModelloFormativo"(codMeccanografico,Indirizzo)
                  ON UPDATE CASCADE ON DELETE NO ACTION

)
```

CREATE TABLE "StudenteUniversitario"

```
(
      Studente char(16) PRIMARY KEY
            REFERENCES "Studente"(CodFiscale)
```

```
                ON UPDATE NO ACTION ON DELETE NO ACTION,
      MediaUltimoAnnoSup numeric NOT NULL,
      votoEsameStato smallint NOT NULL,
      votoCondotta smallint NOT NULL,
      Matricola integer NOT NULL

)
```

CREATE TABLE "Tutor"

```
(

      CodTutor integer,
      Categoria char(1),
      Cognome varchar(30) NOT NULL,
      Nome varchar(30) NOT NULL,
      PRIMARY KEY (CodTutor ,Categoria)

)
```

CREATE TABLE "PFI"

```
(

      CodPFI integer PRIMARY KEY,
      MinutiDaSvolgere smallint NOT NULL,
      QuantitaBorsa numeric (8,2) NOT NULL DEFAULT 0,
      SpeseTot numeric (8,2) NOT NULL DEFAULT 0,
      CodTutor integer NOT NULL,
      MinutiTot smallint DEFAULT 0,
      Categoria char(1) NOT NULL,
      FOREIGN KEY (CodTutor,Categoria)
            REFERENCES "Tutor"(CodTutor,Categoria)
                  ON UPDATE CASCADE ON DELETE NO ACTION
)
```

CREATE TABLE "AreaSpese"

```
(
```

```
      CodCategoria varchar(3) PRIMARY KEY,
      Categoria varchar(20),
      UNIQUE (Categoria)
)
```

CREATE TABLE "Spese"

```
(
      CodSpesa integer PRIMARY KEY,
      Quantita numeric(8,2) NOT NULL DEFAULT 0,
      AreaSpese varchar (3) NOT NULL
            REFERENCES "AreaSpese"(CodCategoria)
                  ON UPDATE CASCADE ON DELETE NO ACTION,
      PFI integer NOT NULL
            REFERENCES "PFI"(CodPFI)
                  ON UPDATE CASCADE ON DELETE NO ACTION

)
```

CREATE TABLE "AnnualitaSecondaria"

```
(
      Studente character(16) ,
      CodMeccanografico char(10),
      Indirizzo char(4),
      Anno smallint,
      ISEE numeric(8,2) NOT NULL,
      Media numeric(4,2) default 0 NOT NULL,
      Attivo char(2) NOT NULL,
      VotoCondotta smallint default 0 NOT NULL,
      PFI integer NOT NULL,
      Tutor integer NOT NULL,
      Categoria char(1) NOT NULL,
      PRIMARY KEY (Studente , CodMeccanografico , Indirizzo , Anno),
      FOREIGN KEY (Studente, CodMeccanografico , Indirizzo)
            REFERENCES "StudenteScuolaSecondaria"(Studente,
            CodMeccanografico,Indirizzo)
```

```
            ON DELETE NO ACTION ON UPDATE CASCADE,
      FOREIGN KEY (Tutor, Categoria)
            REFERENCES "Tutor" (codTutor, categoria)
                  ON DELETE NO ACTION ON UPDATE CASCADE,
      FOREIGN KEY (PFI)
            REFERENCES "PFI" (codPFI)
                  ON DELETE NO ACTION ON UPDATE CASCADE

)
```

CREATE TABLE "AnnualitaUniversitaria"

```
(
      Studente character(16)
            REFERENCES "StudenteUniversitario"(Studente)
                  ON DELETE NO ACTION  ON UPDATE NO ACTION,
      Anno smallint,
      ISEE numeric(8,2) NOT NULL,
      CFUTot smallint,
      Media numeric(4,2),
      Attivo char(2) NOT NULL,
      PFI integer NOT NULL,
      PRIMARY KEY (Studente , Anno),
      FOREIGN KEY (PFI)
            REFERENCES "PFI" (codPFI)
                  ON DELETE NO ACTION ON UPDATE CASCADE

)
```

CREATE TABLE "EsamiSuperati"

```
(
      CodEsame varchar(4) PRIMARY KEY ,
      Nome varchar(70) NOT NULL,
      CFU smallint NOT NULL,
      Voto smallint NOT NULL,
      Studente char(16) ,
```

```
      Anno smallint ,
      FOREIGN KEY (Studente, Anno)
            REFERENCES "AnnualitaUniversitaria"(Studente , Anno)
                  ON DELETE NO ACTION   ON UPDATE NO ACTION

)
```

CREATE TABLE "Supervisione"

```
(
      CodTutor integer ,
      Categoria char(1),
      Attivita varchar(5)
            REFERENCES "ElencoAttivita"(CodAttivita)
                  ON DELETE NO ACTION   ON UPDATE CASCADE,
      PRIMARY KEY (CodTutor,Categoria,Attivita),
      FOREIGN KEY (CodTutor,Categoria)
            REFERENCES "Tutor"(CodTutor,Categoria)
                  ON DELETE NO ACTION   ON UPDATE CASCADE

)
```

CREATE TABLE "AttivitaSvolte"

```
(
PFI integer
      REFERENCES "PFI"(CodPFI)
            ON DELETE NO ACTION   ON UPDATE NO ACTION,
Quantita smallint,
Attivita varchar(5)
      REFERENCES "ElencoAttivita"(CodAttivita)
            ON DELETE NO ACTION   ON UPDATE CASCADE,
PRIMARY KEY (PFI,Attivita)




)
```

CREATE TABLE "Indirizzo"

```
(
      CodIndirizzo char(4) PRIMARY KEY,
      Descrizione varchar(65) NOT NULL,
      UNIQUE (Descrizione)

)
```

Implementazione dei vincoli

Gli esami superati possono avere una votazione compresa fra 18 e 30 e quantità di CFU associati da un minimo di 1:

```
ALTER TABLE  "EsamiSuperati"
ADD CHECK ( "EsamiSuperati".Voto >=18
           AND "EsamiSuperati".Voto <= 30
           AND "EsamiSuperati".CFU >= 1)
```

Gli studenti universitari che partecipano al programma avranno un voto in condotta che varia da 8 a 10, la media dell'ultimo anno di superiori tra 7 e 10 e il voto all'esame di stato tra 80 e 101:

```
ALTER TABLE  "StudenteUniversitario"
ADD CHECK ( "StudenteUniversitario".VotoCondotta >=8
           AND "StudenteUniversitario".VotoCondotta <= 10
           AND "StudenteUniversitario".MediaUltimoAnnoSup >= 7
           AND "StudenteUniversitario".MediaUltimoAnnoSup <= 10
           AND "StudenteUniversitario".VotoEsameStato>= 80
           AND "StudenteUniversitario".VotoEsameStato <= 101)
```

Gli studenti di scuole secondarie che partecipano al programma avranno un voto in condotta che varia da 8 a 10 e la media dell'anno scolastico tra 7 e 10:

```
ALTER TABLE  "StudenteScuolaSecondaria"
ADD CHECK ( "StudenteScuolaSecondaria".VotoCondotta >=8
           AND "StudenteScuolaSecondaria".VotoCondotta <= 10
           AND "StudenteScuolaSecondaria".Media >= 7
           AND "StudenteScuolaSecondaria".Media <= 10)
```

Le attività avranno un minutaggio superiore a 0 per ogni attività:

```
ALTER TABLE  "ElencoAttivita"
ADD CHECK ("ElencoAttivita".MinutiEquiv > 0)
```

Uno studente che partecipa al programma in un determinato anno come studente universitario non può parteciparvi anche come studente di scuola secondaria (e viceversa):

```
CREATE OR REPLACE FUNCTION
public."AnnualitaUniversitariaVincolo"(IN studente character,IN
anno smallint)
    RETURNS boolean
```

```
    LANGUAGE 'plpgsql'
    VOLATILE
    PARALLEL UNSAFE
    COST 100

AS $BODY$
BEGIN
IF ((studente,anno) IN
(SELECT "AnnualitaSecondaria".Studente ,
"AnnualitaSecondaria".Anno
FROM "AnnualitaSecondaria","AnnualitaUniversitaria"
)
)

THEN RETURN false;
ELSE RETURN true;
END IF;
end;
$BODY$;



ALTER TABLE "AnnualitaUniversitaria"
ADD CHECK
("AnnualitaUniversitariaVincolo"("AnnualitaUniversitaria".studente
,"AnnualitaUniversitaria".anno))
```

Uno studente che partecipa al programma in un determinato anno come studente universitario non può parteciparvi anche come studente di scuola secondaria (e viceversa):

```
CREATE OR REPLACE FUNCTION
public."Annualita_Secondaria_vincolo"(IN studente character,IN
anno smallint)
RETURNS boolean
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100

AS $BODY$
```

```
BEGIN
IF ( (studente,anno) IN
(SELECT "AnnualitaUniversitaria".Studente ,
"AnnualitaUniversitaria".Anno
FROM "AnnualitaSecondaria","AnnualitaUniversitaria")
)

THEN RETURN false;
ELSE RETURN true;
END IF;
end;
$BODY$;

ALTER TABLE "AnnualitaSecondaria"
ADD CHECK
("Annualita_Secondaria_vincolo"("AnnualitaSecondaria".studente,"An
nualitaSecondaria".anno))
```

codattivita in ElencoAttivita deve iniziare con 'ATT'

```
ALTER TABLE public."ElencoAttivita"
      ADD CONSTRAINT AttivitaATT_check CHECK (
       "ElencoAttivita".codattivita LIKE 'ATT%'
       );
```

tipologia in ElencoAttivita deve essere 'Presenza' o 'Online'

```
ALTER TABLE public."ElencoAttivita"
      ADD CONSTRAINT tipologiaP_O_check CHECK (
       "ElencoAttivita".tipologia = 'Presenza' or
"ElencoAttivita".tipologia = 'Online'
       );
```

Il Codcomune in Comune deve iniziale con COM

```
ALTER TABLE public."Comune"
      ADD CONSTRAINT comuneCOM_check CHECK (
       "Comune ".codcomune  LIKE 'COM%'
       );
```

Il categoria in Supervisione può essere G o O

```
ALTER TABLE public."Supervisione"
      ADD CONSTRAINT categoriatutorG_O_check CHECK (
      "Supervisione".categoria = 'G' or  "Supervisione".categoria
= 'O'
      );
```

Il codesame in EsamiSuperati deve iniziale con E

```
ALTER TABLE public."EsamiSuperati"
      ADD CONSTRAINT codiceE_check CHECK (
       "EsamiSuperati".codesame LIKE 'E%'
       );
```

Il categoria in PFI deve essere 'G'

```
ALTER TABLE public."PFI"
      ADD CONSTRAINT categoriatutorG_check CHECK (
       "PFI".categoria = 'G'
       );
```

Il categoria in Tutor può essere ‘G’ o ‘O’ o ‘M’

```
ALTER TABLE public."Tutor"
      ADD CONSTRAINT categoriatutorG_O_M_check CHECK (
      "Tutor".categoria = 'G' or  "Tutor".categoria = 'O' or
"Tutor".categoria = 'M'
      );
```

Il sesso in Studente deve essere 'M' o ‘F’

```
ALTER TABLE public."Studente"
      ADD CONSTRAINT categoriatutorG_check CHECK (
       "Studente".sesso = 'M' or  "Studente".sesso = 'F'
       );
```

L’attributo Quantita in AttivitaSvolte deve essere maggiore di zero:

```
ALTER TABLE "AttivitaSvolte"
ADD CHECK ("AttivitaSvolte".quantita > 0)
```

L’attributo Quantita in Spese deve essere maggiore di zero:

```
ALTER TABLE "Spese"
ADD CHECK ("Spese".quantita > 0)
```

Il CodTutor in AnnualitaSecondaria deve essere 'M'

```
ALTER TABLE public."AnnualitaSecondaria"
      ADD CONSTRAINT categoriatutorM_check CHECK (
       "AnnualitaSecondaria".Categoria = 'M'
       );
```

codcategoria in AreaSpese deve iniziare con S

```
ALTER TABLE public."AreaSpese"
      ADD CONSTRAINT areaSpese_S_check CHECK (
      "AreaSpese".codcategoria LIKE 'S%'
      );
```

isee in AnnualitaUniversitaria deve essere positivo

```
ALTER TABLE public."AnnualitaUniversitaria"
      ADD CONSTRAINT Isee_not_neg_check CHECK (
      "AnnualitaUniversitaria".isee >=0
      );
```

isee in AnnualitaSecondaria deve essere positivo

```
ALTER TABLE public."AnnualitaSecondaria"
      ADD CONSTRAINT Isee_not_neg_check CHECK (
      "AnnualitaSecondaria".isee >=0
      );
```

Attivo in AnnualitaUniversitaria deve essere SI o NO

```
ALTER TABLE public."AnnualitaUniversitaria"
      ADD CONSTRAINT AttivoY_N_check CHECK (
      "AnnualitaUniversitaria".attivo = 'SI' or
      "AnnualitaUniversitaria".attivo = 'NO'
      );
```

Data iniziale di ElencoAttivita deve essere minore di Data finale

```
ALTER TABLE "ElencoAttivita"
ADD CHECK ("ElencoAttivita".datainiziale <=
"ElencoAttivita".datafinale)
```

Attivo in AnnualitaSecondaria deve essere SI o NO

```
ALTER TABLE public."AnnualitaSecondaria"
      ADD CONSTRAINT AttivoY_N_check CHECK (
      "AnnualitaSecondaria".attivo = 'SI' or
      "AnnualitaSecondaria".attivo = 'NO'
      );
```

Le attivita in presenza possono essere fatte solo una volta

```
CREATE OR REPLACE FUNCTION public."Attivitapresenza_vincolo"(IN
Attivita varchar, IN Quantita smallint)
```

```
RETURNS boolean
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
     COST 100

AS $BODY$
BEGIN
IF ( Quantita  <> '1'
     and (Attivita)
     IN
     (select "ElencoAttivita".codattivita
     from "ElencoAttivita"
     WHERE  "ElencoAttivita".tipologia='Presenza')
)


THEN RETURN false;
ELSE RETURN true;
END IF;
end;
$BODY$;

ALTER TABLE "AttivitaSvolte"
ADD CHECK
("Attivitapresenza_vincolo"("AttivitaSvolte".attivita,"AttivitaSvo
lte".quantita))
```

I Tutor aggiunti con lo stesso Codice Tutor devono avere lo stesso Nome e Cognome:

```
CREATE OR REPLACE FUNCTION public."vincoloTutor"(IN addedtutor
integer,IN addedcognome character varying,IN addednome character
varying)
RETURNS boolean
LANGUAGE 'plpgsql'

AS $BODY$
BEGIN
```

```
IF ((addednome <> (SELECT "Tutor".nome
                          FROM "Tutor"
                  WHERE "Tutor".codtutor = addedtutor
                  GROUP BY "Tutor".nome))
     or
     (addedcognome <> (SELECT  "Tutor".cognome
                       FROM "Tutor"
                       WHERE "Tutor".codtutor = addedtutor
                       GROUP BY "Tutor".cognome)))

THEN RETURN FALSE;
ELSE RETURN TRUE;
END IF;
end;
$BODY$;

ALTER TABLE "Tutor"
ADD CHECK ("vincoloTutor"("Tutor".codtutor,"Tutor".cognome,
"Tutor".nome))
```

Controllo sulla ridondanza di cfutot:

```
CREATE OR REPLACE FUNCTION public."vincoloStudenteCFU"(IN
studentescelto character,IN annoscelto smallint,IN cfuscelti
smallint)
    RETURNS boolean
    LANGUAGE 'plpgsql'
    VOLATILE
    PARALLEL UNSAFE
    COST 100

AS $BODY$

BEGIN
IF ( (cfuscelti) <> (SELECT sum("EsamiSuperati".cfu)
FROM "EsamiSuperati"
where "EsamiSuperati".studente = studentescelto and
"EsamiSuperati".anno = annoscelto)
 )
```

```
THEN RETURN FALSE;
ELSE RETURN TRUE;
END IF;
end;
$BODY$;

ALTER TABLE "AnnualitaUniversitaria"
ADD CHECK ("vincoloStudenteCFU"("AnnualitaUniversitaria".studente
, "AnnualitaUniversitaria".anno ,
"AnnualitaUniversitaria".CFUtot))
```

Le occorrenze di Supervisione dovranno avere la categoria Tutor che rispetta la relativa tipologia di Attività:

```
CREATE OR REPLACE FUNCTION public."vincoloSupervisione"(IN
codtutor integer,IN categoria character,IN codattivita character
varying)
    RETURNS boolean
    LANGUAGE 'plpgsql'
    VOLATILE
    PARALLEL UNSAFE
    COST 100

AS $BODY$
BEGIN
IF ( (categoria = 'G' AND codattivita IN (
          SELECT "ElencoAttivita".codattivita
          FROM "ElencoAttivita"
          WHERE "ElencoAttivita".tipologia = 'Presenza'))
     OR (categoria = 'O' AND codattivita IN (
          SELECT "ElencoAttivita".codattivita
          FROM "ElencoAttivita"
          WHERE "ElencoAttivita".tipologia = 'Online')))
THEN RETURN TRUE;
ELSE RETURN FALSE;
```

```
END IF;
end;
$BODY$;


ALTER TABLE "Supervisione"
ADD CHECK
("vincoloSupervisione"("Supervisione".codtutor,"Supervisione".cate
goria, "Supervisione".attivita))
```

Controllo sulla ridondanza di spesetot in PFI:

```
CREATE OR REPLACE FUNCTION public."spese_tot_vincolo"(IN spesetot
numeric(8,2), IN codPfi integer )
RETURNS boolean
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100

AS $BODY$
BEGIN
IF ( (spesetot) <>
(SELECT sum("Spese".quantita)
FROM "Spese"
where "Spese".pfi = codpfi)
)

THEN RETURN false;
ELSE RETURN true;
END IF;
end;
$BODY$;
ALTER TABLE "PFI"
ADD CHECK ("spese_tot_vincolo"("PFI".spesetot,"PFI".codpfi))
```

Controllo sulla ridondanza di minutitot in PFI:

```
CREATE OR REPLACE FUNCTION public."minuti_tot_vincolo"(IN
minutitot numeric(8,2), IN codPfi integer )
RETURNS boolean
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100

AS $BODY$
DECLARE minuticheck integer = (select
sum("AttivitaSvolte".quantita * "ElencoAttivita".minutiequiv)
                                        FROM
"AttivitaSvolte","ElencoAttivita"
                                        where
codPfi="AttivitaSvolte".pfi and
"AttivitaSvolte".attivita="ElencoAttivita".codattivita );

BEGIN
IF ( (minutitot) <> (minuticheck))

THEN RETURN false;
ELSE RETURN true;
END IF;
end;
$BODY$;

ALTER TABLE "PFI"
ADD CHECK ("minuti_tot_vincolo"("PFI".minutitot,"PFI".codpfi))
```

Implementazione delle operazioni

OP1: Aggiunta di uno studente

```
CREATE FUNCTION aggiungiStudente
(
	CodFiscale  char(16),
	Nome varchar,
	Cognome varchar,
	DataNascita date,
	Sesso char(1),
	Comune varchar
)
returns void
language plpgsql
as $$
begin

INSERT INTO "Studente"
VALUES (
	CodFiscale,
	Nome,
	Cognome,
	DataNascita,
	Sesso,
	Comune
);

end;
$$;
```

OP2: Aggiunta di un PFI

```
CREATE or replace FUNCTION aggiungiPFI
(
	MinutiDaSvolgere integer,
	QuantitaBorsa numeric (8,2),
	CodTutor integer
)
```

```
returns void
language plpgsql
as $$
     declare codnew integer= (select max("PFI".codPFI)
                                   FROM "PFI")+1;
     declare spesenew numeric(6,2)=0.00;
     declare catnew char(1)='G';
     declare minnew smallint=0;
     begin
          INSERT INTO "PFI"
          VALUES(
               codnew,
               MinutiDaSvolgere,
               QuantitaBorsa ,
               spesenew ,
               CodTutor ,
               catnew,
               minnew

          );
     end;
$$;
```

OP3: Aggiunta di un attività all' elenco attività

```
CREATE FUNCTION aggiungiAttivita
(
     CodAttivita varchar,
     Tipologia varchar,
     DataIniziale date,
     DataFinale date,
     Nome varchar,
     MinutiEquiv smallint,
     Luogo varchar
)
returns void
language plpgsql
as $$
```

```
	begin
		INSERT INTO "ElencoAttivita"
		VALUES(
			CodAttivita,
			Tipologia,
			DataIniziale,
			DataFinale,
			Nome,
			MinutiEquiv,
			Luogo
		);
	end;
$$;
```

OP4: Aggiunta di un'annualità secondaria

```
CREATE OR REPLACE FUNCTION public.aggiungiannualitasecondaria(IN
studente character,IN codmeccanografico character,IN indirizzo
character,IN anno smallint,IN isee numeric,IN attivo character,IN
pfi integer, IN tutor integer , IN categoria character)
	RETURNS void
	LANGUAGE 'plpgsql'
	VOLATILE
	PARALLEL UNSAFE
	COST 100

AS $BODY$
declare medianew numeric(4,2)=0.00;
declare votonew smallint=0;
begin

INSERT INTO "AnnualitaSecondaria"
VALUES(
	Studente ,
	CodMeccanografico,
	Indirizzo,
	Anno,
	ISEE,
```

```
      medianew,
      Attivo,
      votonew,
      PFI,
      tutor,
      categoria
);
end;
$BODY$;
```

OP5: Aggiunta di un'annualità Universitaria

```
CREATE FUNCTION aggiungiAnnualitaUniversitaria
(
      Studente character(16),
      Anno smallint,
      ISEE numeric(8,2),
      Attivo char(2),
      PFI integer

)
returns void
language plpgsql
as $$
      declare medianew numeric(4,2)=0.00;
      declare votonew smallint=0;

      begin
            INSERT INTO "AnnualitaUniversitaria"
            VALUES(
                  Studente,
                  Anno,
                  ISEE,
                  votonew,
                  medianew,
                  Attivo,
                  PFI
```

```
            );
      end;
$$;
```

OP6: Aggiunta di un esame superato da uno studente

```
CREATE OR REPLACE FUNCTION public.aggiungiesamesuperato(
IN ncodesame character varying,
IN nnome character varying,
IN ncfu smallint,
IN nvoto smallint,
IN nstudente character,
IN nanno smallint)
      RETURNS void
      LANGUAGE 'plpgsql'
      VOLATILE
      PARALLEL UNSAFE
      COST 100

AS $BODY$
      DECLARE prodesam integer = (nVoto*nCFU)+ (select
      sum("EsamiSuperati".cfu *"EsamiSuperati".voto)
      from "EsamiSuperati"
      where
      "EsamiSuperati".studente=nstudente
       and "EsamiSuperati".anno=nanno);

      DECLARE    cfu_new integer= nCFU+(select
      sum("EsamiSuperati".cfu)
      from "EsamiSuperati"
      where "EsamiSuperati".Studente=nstudente and
      "EsamiSuperati".anno=nanno );
      DECLARE nuovmed numeric(4,2)= prodesam/cfu_new;
      BEGIN
            INSERT INTO "EsamiSuperati"
            VALUES(
                  nCodEsame,
            nNome,
```

```
            nCFU,
            nVoto,
            nStudente,
            nAnno
                 );
            /*modifica CFUtot e Media*/
            UPDATE "AnnualitaUniversitaria"
            SET   CFUtot  = cfu_new,
                        media = nuovmed
            WHERE
             nStudente = "AnnualitaUniversitaria".Studente
             and
             nanno= "AnnualitaUniversitaria".anno ;


      end;
$BODY$;
```

OP7: Aggiunta di un Tutor

```
CREATE FUNCTION aggiungiTutor(
                        CodTutor integer,
                        Categoria char(1),
                        Cognome varchar,
                        Nome varchar

)
returns void
language plpgsql
as $$
      begin
            INSERT INTO "Tutor"
            VALUES(
                        CodTutor,
                        Categoria,
                        Cognome,
```

```
                    Nome
                );
    end;
$$;
```

OP8: Modifica della residenza di uno studente

```
CREATE or REPLACE FUNCTION modificaResidenzaStudente(
                    oldStudente char(16),
                    newComune varchar
)
returns void
language plpgsql
as $$
    begin
         UPDATE "Studente"
         SET Comune = newComune
        WHERE CodFiscale = oldStudente    ;

    end;
$$;
```

OP9: Aggiornamento di un PFI

```
CREATE OR REPLACE FUNCTION modificaPFI(
          mCodPFI integer,
          mMinutiDaSvolgere smallint,
          mQuantitaBorsa numeric (6,2),
          mCodTutor integer,
          mMinutiTot smallint
)
returns void
language plpgsql
as $$
begin
    UPDATE "PFI"
SET  MinutiDaSvolgere = mMinutiDaSvolgere,
QuantitaBorsa = mQuantitaBorsa,
CodTutor = mCodTutor,
MinutiTot = mMinutiTot
    WHERE "PFI".CodPFI = mCodPFI;
```

```
end;
$$;
```

OP10: Eliminazione di un esame superato da uno studente

```
CREATE OR REPLACE FUNCTION public.eliminasamesuperato(IN ncodesame
character varying)
RETURNS void
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100

AS $BODY$
DECLARE nomestud char(16)=(SELECT "EsamiSuperati".studente FROM
"EsamiSuperati" WHERE "EsamiSuperati".codesame = ncodesame);

DECLARE annostud smallint= (SELECT "EsamiSuperati".anno FROM
"EsamiSuperati" WHERE "EsamiSuperati".codesame = ncodesame) ;

DECLARE prodesam integer =

            (select sum("EsamiSuperati".cfu *"EsamiSuperati".voto)
            from "EsamiSuperati"
            where "EsamiSuperati".studente = nomestud and
            "EsamiSuperati".anno = annostud and
            "EsamiSuperati".codesame<>ncodesame );

      DECLARE    cfu_new integer=
            (select sum("EsamiSuperati".cfu)
            from "EsamiSuperati"
            where "EsamiSuperati".studente = nomestud and
            "EsamiSuperati".anno = annostud and
            "EsamiSuperati".codesame<>ncodesame);

      DECLARE nuovmed numeric(4,2)= prodesam/cfu_new;
      BEGIN
```

```
            DELETE FROM "EsamiSuperati"
            WHERE "EsamiSuperati".codesame = ncodesame;

            /*modifica CFUtot e Media*/
            UPDATE "AnnualitaUniversitaria"
            SET   CFUtot  = cfu_new,
                  media = nuovmed
            WHERE nomestud = "AnnualitaUniversitaria".Studente
             and
             annostud = "AnnualitaUniversitaria".anno ;


     end;
$BODY$;
```

OP11: Stampa gli studenti universitari che rispettano gli obiettivi di merito in un dato anno

```
CREATE OR REPLACE FUNCTION
stampa_universitari_anno_rispettano_merito(anno_sel integer)
RETURNS TABLE (studente character(16),media numeric, cfutot
smallint)
LANGUAGE 'plpgsql'

AS $BODY$
BEGIN
return query
(select
"AnnualitaUniversitaria".studente,"AnnualitaUniversitaria".media,"
AnnualitaUniversitaria".cfutot
from "AnnualitaUniversitaria"
where anno_sel="AnnualitaUniversitaria".anno AND
"AnnualitaUniversitaria".media>=27.00 AND
"AnnualitaUniversitaria".cfutot>15);
end
$BODY$;
```

OP12: Stampa i pfi che rispettano i minuti di partecipazione in un dato anno

```
CREATE OR REPLACE FUNCTION
stampa_PFI_anno_rispettano_minuti(anno_sel integer)
RETURNS TABLE (codPFI integer,minuti_da_svolgere
numeric,minuti_fatti smallint)
LANGUAGE 'plpgsql'

AS $BODY$
     BEGIN
          return query
          (select
          "PFI".codpfi,"PFI".minutidasvolgere,"PFI".minutitot
          from "PFI","AnnualitaUniversitaria"
          where anno_sel="AnnualitaUniversitaria".anno AND
          "AnnualitaUniversitaria".pfi="PFI".codpfi AND
          "PFI".minutitot>="PFI".minutidasvolgere);
     end
$BODY$;
```

OP13: Stampa il valore medio della media dei voti degli studenti delle scuole secondarie in un dato anno

```
CREATE OR REPLACE FUNCTION
stampa_media_delle_medie_anno_secondaria(anno_sel integer)
RETURNS TABLE (mediaMedie numeric (4,2))
LANGUAGE 'plpgsql'

AS $BODY$
     BEGIN
          return query
          (select ROUND((avg("AnnualitaSecondaria".media)),2)
          from "AnnualitaSecondaria"
          where anno_sel="AnnualitaSecondaria".anno);
     end
$BODY$;
```

OP14: Stampa gli studenti universitari che hanno superato da un numero minimo di esami ad un numero massimo di esami in un dato anno

```
CREATE OR REPLACE FUNCTION
stampa_Universitari_anno_minimo_massimo_esamiSuperati(anno_sel
integer,esamimin integer, esamimax integer)
RETURNS TABLE (studente character(16), numEsami bigint)
LANGUAGE 'plpgsql'

AS $BODY$
     BEGIN
          return query
          (select "AnnualitaUniversitaria".studente, count(*) as
          numEsami
          from "AnnualitaUniversitaria","EsamiSuperati"
          where anno_sel="EsamiSuperati".anno and
          "EsamiSuperati".studente="AnnualitaUniversitaria".stude
          nte
          group by "AnnualitaUniversitaria".studente
          having count(*)>=esamimin and count(*)<=esamimax);
     end
$BODY$;
```

OP15: Stampa i Tutor che appartengono a più categorie di Tutor

```
CREATE OR REPLACE FUNCTION
stampa_tutor_appartengono_più_categorie_()
RETURNS TABLE (codtutor integer,nome character varying(30),cognome
character varying(30))
LANGUAGE 'plpgsql'

AS $BODY$
     BEGIN
          return query
          (select "Tutor".codtutor,"Tutor".nome,"Tutor".cognome
          from "Tutor"
          where "Tutor".codtutor in  (select "Tutor".codtutor
          from "Tutor"
          group by "Tutor".codtutor
          having count(*)>1)
```

```
            group by "Tutor".codtutor,"Tutor".nome,"Tutor".cognome
            );
      end
$BODY$;
```

OP16: Stampa i Tutor che appartengono a una sola categoria di tutor

```
CREATE OR REPLACE FUNCTION stampa_tutor_appartiene_una_categoria()
RETURNS TABLE (codtutor integer,nome character varying(30),cognome
character varying(30))
LANGUAGE 'plpgsql'

AS $BODY$
      BEGIN
            return query
            (select "Tutor".codtutor,"Tutor".nome,"Tutor".cognome
            from "Tutor"
            where "Tutor".codtutor in  (select "Tutor".codtutor
                                                from "Tutor"
                                                group by
            "Tutor".codtutor
                                                having count(*)=1)
            group by "Tutor".codtutor,"Tutor".nome,"Tutor".cognome
            );
      end
$BODY$;
```

OP17: Stampa il numero di partecipanti attivi della scuola secondaria in un dato anno

```
CREATE OR REPLACE FUNCTION
stampa_numero_partecipanti_anno_secondaria(anno_sel integer)
RETURNS TABLE (numPartecipantiSecondaria bigint)
LANGUAGE 'plpgsql'

AS $BODY$
      BEGIN
            return query
            (select count(*) as numPartecipanti
            from "AnnualitaSecondaria"
```

```
            where anno_sel="AnnualitaSecondaria".anno and
            "AnnualitaSecondaria".attivo='SI');
      end
$BODY$;
```

OP18: Stampa il numero di partecipanti attivi universitari in un dato anno

```
CREATE OR REPLACE FUNCTION
stampa_numero_partecipanti_anno_Universitaria(anno_sel integer)
RETURNS TABLE (numPartecipantiSecondaria bigint)
LANGUAGE 'plpgsql'

AS $BODY$
      BEGIN
            return query
            (select count(*) as numPartecipanti
            from "AnnualitaUniversitaria"
            where anno_sel="AnnualitaUniversitaria".anno and
            "AnnualitaUniversitaria".attivo='SI');
      end
$BODY$;
```

OP19: Per ogni scuola stampa quanti studenti hanno partecipato al programma in un dato anno

```
CREATE OR REPLACE FUNCTION
stampa_scuole_studenti_partecipanti_anno(anno_sel integer)
RETURNS TABLE (codMeccanografico character, nome character varying
(30))
LANGUAGE 'plpgsql'

AS $BODY$
      BEGIN
            return query
            (select "Scuola".codmeccanografico,"Scuola".nome
            from "AnnualitaSecondaria","Scuola"
            where anno_sel="AnnualitaSecondaria".anno and
            "AnnualitaSecondaria".attivo='SI' and
            "AnnualitaSecondaria".codmeccanografico="Scuola".codmec
            canografico
```

```
          group by "Scuola".codmeccanografico);
     end
$BODY$;
```

OP20: Stampa il numero di attività per ogni studente in un dato anno

```
CREATE OR REPLACE FUNCTION
public."Stampa_Num_Attivita_Studente"(IN annoscelto smallint)
RETURNS TABLE(studente character, numattivita bigint)
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100
ROWS 1000

AS $BODY$
     BEGIN
          return query (
          (
          SELECT "AnnualitaUniversitaria".studente as studente,
          Count(*) as numAtt
          FROM "AnnualitaUniversitaria" , "PFI" ,
          "AttivitaSvolte" as "AttS"
          WHERE "PFI".codPFI = "AnnualitaUniversitaria".PFI
          AND "AttS".pfi = "PFI".codPFI
          AND "AnnualitaUniversitaria".Anno = annoscelto
          group by "AnnualitaUniversitaria".studente
          )
               UNION
          (
          SELECT "Sec".studente as studente, Count(*) as numAtt
          FROM "AnnualitaSecondaria" as "Sec" , "PFI" ,
          "AttivitaSvolte" as "AttS"
          WHERE "PFI".codPFI = "Sec".PFI AND "AttS".pfi =
          "PFI".codPFI AND "Sec".Anno = annoscelto
          GROUP BY "Sec".studente
          )

          );
```

```
    end
$BODY$;
```

OP21: Stampa il numero di CFU ottenuto da ogni studente universitario in un dato anno

```
CREATE OR REPLACE FUNCTION public."CFU_Studente_Anno"(IN
annoscelto smallint)
RETURNS TABLE(studente character, numcfu smallint)
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100
ROWS 1000

AS $BODY$
BEGIN
RETURN QUERY (
(
SELECT Uni.studente as Studente, Uni.CFUTot as NumCFU
FROM "AnnualitaUniversitaria" as Uni
WHERE Uni.Anno = annoscelto
)
);
end
$BODY$;
```

OP22: Modifica lo stato di attività di uno studente universitario

```
CREATE OR REPLACE FUNCTION
public."Modifica_Stato_StudenteUniversitario"(IN studentescelto
character,IN annocorrente smallint,IN nuovoattivo character)
    RETURNS void
    LANGUAGE 'plpgsql'
    VOLATILE
    PARALLEL UNSAFE
    COST 100

AS $BODY$
```

```
BEGIN
IF ( (studentescelto,annocorrente) IN (
     SELECT Uni.studente as studente, anno as annocorrente
     FROM "AnnualitaUniversitaria" as Uni))
THEN UPDATE "AnnualitaUniversitaria"
SET attivo = nuovoattivo
WHERE ("AnnualitaUniversitaria".studente = studentescelto AND
"AnnualitaUniversitaria".anno = annocorrente);

END IF;

end;
$BODY$;
```

OP23: Modifica dei CFU ottenuti da uno studente universitario in un dato esame

```
CREATE OR REPLACE FUNCTION public.Modifica_Esame_CFU(IN ncodesame
character varying,IN ncfu smallint)
RETURNS void
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100

AS $BODY$
DECLARE nomestud char(16)=(SELECT "EsamiSuperati".studente FROM
"EsamiSuperati" WHERE "EsamiSuperati".codesame = ncodesame);

DECLARE annostud smallint= (SELECT "EsamiSuperati".anno FROM
"EsamiSuperati" WHERE "EsamiSuperati".codesame = ncodesame) ;

DECLARE prodesam integer =
(select sum("EsamiSuperati".cfu *"EsamiSuperati".voto)
from "EsamiSuperati"
where "EsamiSuperati".studente = nomestud and "EsamiSuperati".anno
= annostud and "EsamiSuperati".codesame<>ncodesame ) +
```

```
(ncfu*(select "EsamiSuperati".voto
      from "EsamiSuperati"
      where "EsamiSuperati".codesame=ncodesame));

DECLARE    cfu_new integer= ncfu+
(select sum("EsamiSuperati".cfu)
from "EsamiSuperati"
where "EsamiSuperati".studente = nomestud and "EsamiSuperati".anno
= annostud and "EsamiSuperati".codesame<>ncodesame);

DECLARE nuovmed numeric(4,2)= prodesam/cfu_new;

BEGIN

/*modifica CFUtot e Media*/

UPDATE "EsamiSuperati"
SET   CFU   = ncfu
WHERE "AnnualitaUniversitaria".codesame = ncodesame ;

UPDATE "AnnualitaUniversitaria"
SET   CFUtot   = cfu_new,
          media = nuovmed
WHERE nomestud = "AnnualitaUniversitaria".Studente
 and
 annostud = "AnnualitaUniversitaria".anno ;



end;
$BODY$;
```

OP24: Modifica lo stato di attività di uno studente secondario

```
CREATE OR REPLACE FUNCTION
public."Modifica_Stato_StudenteSecondaria"(IN studentescelto
character,IN annocorrente smallint,IN codmeccanograficoscelto
character,IN indirizzoscelto character,IN nuovoattivo character)
    RETURNS void
    LANGUAGE 'plpgsql'
```

```
    VOLATILE
    PARALLEL UNSAFE
    COST 100

AS $BODY$
BEGIN

IF( (studentescelto,annocorrente,codmeccanograficoscelto,
indirizzoscelto) IN (
     SELECT Sec.studente as studente, Sec.anno as annocorrente  ,
Sec.codmeccanografico , Sec.indirizzo
     FROM "AnnualitaSecondaria" as Sec)
     )
THEN UPDATE "AnnualitaSecondaria"
SET attivo = nuovoattivo
WHERE ("AnnualitaSecondaria".studente = studentescelto AND
"AnnualitaSecondaria".anno = annocorrente AND
"AnnualitaSecondaria".codmeccanografico AND
"AnnualitaSecondaria".indirizzo);
END IF;

end;
$BODY$;
```

OP25: stampa quanti PFI ha approvato ogni Tutor di Gestione

```
CREATE OR REPLACE FUNCTION
public.numero_pfi_approvati_per_Tutor(IN codicetutor integer)
RETURNS TABLE(tutor integer, nome character varying, numpfi
bigint)
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100
ROWS 1000

AS $BODY$
BEGIN
```

```
return query (
(
select "Tutor".codtutor, "Tutor".nome , Count(*)
from "Tutor" , "PFI"
where "PFI".codtutor = "Tutor".codtutor AND "PFI".categoria =
"Tutor".categoria AND "Tutor".codtutor = codiceTutor
group by "Tutor".codtutor, "Tutor".nome
)

);
end
$BODY$;
```

OP26: Stampa il numero di studenti in un dato anno che hanno un residuo inferiore a 100€ nella borsa di studio

```
CREATE OR REPLACE FUNCTION
public.Studenti_Con_Residuo_minore_100(IN annoscelto smallint)
    RETURNS TABLE(codpfi integer, residuo numeric)
    LANGUAGE 'plpgsql'
    VOLATILE
    PARALLEL UNSAFE
    COST 100    ROWS 1000

AS $BODY$
BEGIN
return query (
(


     (
SELECT  pfi.codpfi as codpfi, (pfi.quantitaborsa - pfi.spesetot)
as Residuo
FROM "AnnualitaUniversitaria" as Uni , "PFI" as pfi
WHERE Uni.Anno = annoscelto AND pfi.codpfi = Uni.pfi AND
(pfi.quantitaborsa - pfi.spesetot) <= 100
     )
     UNION
     (
```

```
SELECT pfi.codpfi as codpfi, (pfi.quantitaborsa - pfi.spesetot) as
Residuo
FROM "AnnualitaSecondaria" as Sec , "PFI" as pfi
WHERE Sec.Anno = annoscelto AND pfi.codpfi = Sec.pfi AND
(pfi.quantitaborsa - pfi.spesetot) <= 100
     )
)
);
end
$BODY$;
```

OP27: Stampa media universitario per ogni studente in un dato anno

```
CREATE OR REPLACE FUNCTION public.Stampa_Media_Universitario(IN
annoscelto smallint)
RETURNS TABLE(studente character, media numeric)
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100
ROWS 1000

AS $BODY$
BEGIN
RETURN QUERY (

SELECT Uni.studente , Uni.media
FROM "AnnualitaUniversitaria" as Uni
WHERE Uni.Anno = annoscelto

);
end
$BODY$;
```

OP28: Stampa per ogni categoria quanti soldi sono stati spesi complessivamente

```
CREATE OR REPLACE FUNCTION public.Stampa_Spese_Per_Categoria(IN
annoscelto smallint)
```

```
RETURNS TABLE(codcategoria character varying, categoria character
varying, soldispesi numeric)
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100    ROWS 1000

AS $BODY$
BEGIN
RETURN QUERY (

SELECT area.codcategoria , area.categoria , SUM(spese.quantita)
FROM "AreaSpese" as area , "Spese" as spese
WHERE spese.areaspese = area.codcategoria
GROUP BY area.codcategoria , area.categoria

);
end
$BODY$;
```

OP29: Aggiunta di una spesa

```
CREATE OR REPLACE FUNCTION public."Aggiungi_Spesa"(IN quantita
numeric,IN areaspese character varying,IN pfi integer)
RETURNS void
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100

AS $BODY$
declare codnew integer= (SELECT max("Spese".codspesa)
                         FROM "Spese")+1;
declare newspes numeric(8,2)= quantita + (SELECT "PFI".spesetot
                                          FROM "PFI"
                                          WHERE "PFI".codpfi=pfi
);
BEGIN
INSERT INTO "Spese"
VALUES (codnew ,quantita,areaspese, pfi);
```

```
UPDATE  "PFI"
SET spesetot= newspes
WHERE "PFI".codpfi=pfi ;


end;
$BODY$;
```

OP30: Aggiunta di un'attività svolta

```
CREATE OR REPLACE FUNCTION public.aggiunta_attivita_svolta(IN pfi
integer,IN quantita smallint,IN attivita character varying)
    RETURNS void
    LANGUAGE 'plpgsql'
    VOLATILE
    PARALLEL UNSAFE
    COST 100

AS $BODY$
DECLARE
minutinuov integer = quantita*(select "ElencoAttivita".minutiequiv
                            from "ElencoAttivita"
                            where
"ElencoAttivita".codattivita=attivita) + (select "PFI".minutitot
                                                    from "PFI"
                                                    where
"PFI".codpfi=pfi);
BEGIN
IF ( pfi IN (
SELECT "PFI".codpfi
FROM "PFI") AND attivita IN(
   SELECT "ElencoAttivita".codattivita
   From "ElencoAttivita"))
THEN INSERT INTO "AttivitaSvolte"
VALUES (pfi,quantita,attivita) ;
UPDATE  "PFI"
       set minutitot= minutinuov
```

```
        where "PFI".codpfi=pfi ;

END IF;
end;
$BODY$;
```

OP31: Aggiunta della supervisione da parte di un tutor di un'attività

```
CREATE OR REPLACE FUNCTION public."Aggiungi_Supervisione"(IN
codtutor integer,IN categoria character,IN codattivita character
varying)
    RETURNS void
    LANGUAGE 'plpgsql'
    VOLATILE
    PARALLEL UNSAFE
    COST 100

AS $BODY$
BEGIN

INSERT INTO "Supervisione"
VALUES(codtutor,categoria,codattivita);

end;
$BODY$;
```

OP32: Modifica della media e del voto finale di un'Annualità Secondaria

```
CREATE OR REPLACE FUNCTION modifica_annualita_secondaria(
nstudente char(16),
ncodmec char(10),
nanno smallint,
nindi char(4),
```

```
nmedia numeric (4,2),
nvotocondotta smallint
)
returns void
language plpgsql
as $$
begin
     UPDATE "AnnualitaSecondaria"
SET
media = nmedia,
votocondotta = nvotocondotta

WHERE "AnnualitaSecondaria".studente = nstudente and
"AnnualitaSecondaria".codmeccanografico = ncodmec AND
"AnnualitaSecondaria".indirizzo = nindi
     and "AnnualitaSecondaria".anno = nanno;

end;
$$;
```

Esecuzione delle operazioni principali

OP2: Aggiunta di un PFI

Supponendo di partire dalla tabella mostrata di “PFI” si applica la funzione per l’aggiunta di un nuovo PFI.

Data Output

	codpfi [PK] integer	minutidasvolgere smallint	quantitaborsa numeric (8,2)	spesetot numeric (8,2)	codtutor integer	categoria character (	minutitot smallint
422	422	1500	3600.00	165.60	3	G	195
423	423	1200	3600.00	180.00	11	G	195
424	424	900	3000.00	192.00	11	G	135
425	425	1100	3500.00	161.00	2	G	60
426	426	1000	4800.00	336.00	12	G	195
427	427	1500	3500.00	234.50	12	G	195
428	428	1100	3500.00	224.00	3	G	195
429	429	950	4800.00	297.60	3	G	195
430	430	1000	3000.00	171.00	12	G	195
431	431	800	3000.00	171.00	13	G	195
432	432	1400	3000.00	228.00	11	G	135
433	433	1500	3000.00	135.00	11	G	195
434	434	800	3000.00	210.00	12	G	195
435	435	1400	3000.00	201.00	11	G	195
436	436	1200	3600.00	270.00	1	G	135
437	437	1500	4800.00	398.40	12	G	195
438	438	1200	3000.00	174.00	1	G	195
439	439	900	3500.00	255.50	3	G	195
440	440	900	3000.00	171.00	2	G	195
441	441	1400	3600.00	277.20	3	G	195
442	442	700	3500.00	231.00	1	G	195
443	443	1400	3600.00	248.40	12	G	195
444	444	900	3000.00	186.00	2	G	195
445	445	1200	3000.00	156.00	3	G	0
446	446	900	4800.00	364.80	11	G	195

Si vogliono inserire i dati del nuovo PFI invocando la funzione “aggiungiPFI”.

```
SELECT (aggiungiPFI('1000','5000','11'));
```

Data Output

	codpfi [PK] integer	minutidasvolgere smallint	quantitaborsa numeric (8,2)	spesetot numeric (8,2)	codtutor integer	categoria character	minutitot smallint
423	423	1200	3600.00	180.00	11	G	195
424	424	900	3000.00	192.00	11	G	135
425	425	1100	3500.00	161.00	2	G	60
426	426	1000	4800.00	336.00	12	G	195
427	427	1500	3500.00	234.50	12	G	195
428	428	1100	3500.00	224.00	3	G	195
429	429	950	4800.00	297.60	3	G	195
430	430	1000	3000.00	171.00	12	G	195
431	431	800	3000.00	171.00	13	G	195
432	432	1400	3000.00	228.00	11	G	135
433	433	1500	3000.00	135.00	11	G	195
434	434	800	3000.00	210.00	12	G	195
435	435	1400	3000.00	201.00	11	G	195
436	436	1200	3600.00	270.00	1	G	135
437	437	1500	4800.00	398.40	12	G	195
438	438	1200	3000.00	174.00	1	G	195
439	439	900	3500.00	255.50	3	G	195
440	440	900	3000.00	171.00	2	G	195
441	441	1400	3600.00	277.20	3	G	195
442	442	700	3500.00	231.00	1	G	195
443	443	1400	3600.00	248.40	12	G	195
444	444	900	3000.00	186.00	2	G	195
445	445	1200	3000.00	156.00	3	G	0
446	446	900	4800.00	364.80	11	G	195
447	447	1000	5000.00	0.00	11	G	0

OP10: Eliminazione di un esame superato

Si vuole eliminare un esame relativo ad un esame superato da uno studente universitario.
Si sceglie di eliminare l'esame "E99".

Data Output | Explain | Messages | Notifications

	codesame [PK] character varying (4)	nome character varying (70)	cfu smallint	voto smallint	studente character (16)	anno smallint
280	E85	Istologia C.I.	8	25	PCHCLT00H52A56...	2019
281	E86	Anatomia, istologia e fisiolo...	6	27	DCLRRT00R19D61...	2019
282	E87	Informatica	3	24	PCHCLT00H52A56...	2019
283	E88	Chimica analitica	12	26	FBBGDI00D61A851X	2019
284	E89	Chimica generale e inorganica	6	19	RSSSMN00S30D61...	2019
285	E9	TEORIE DEI PROCESSI COM...	12	25	MSCCHR00P63D6...	2019
286	E90	Antropologia di genere	12	27	MSCCHR00P63D6...	2019
287	E91	Laboratorio greco 2	3	29	SMLLSS00M67E20...	2019
288	E92	Linguistica generale	12	25	RLNDNS00T19F03...	2019
289	E93	Fisica	5	22	PCHCLT00H52A56...	2019
290	E94	Statistica I	9	20	CLMGLI01B54D612X	2019
291	E95	Scienze umane	4	19	PCHCLT00H52A56...	2019
292	E96	Psicologia Sociale	9	18	PLNTMS00R20I726P	2019
293	E97	Microbiologia	1	21	SGRNCHR00B68A8...	2019
294	E98	Chimica Generale e Inorgani...	9	21	SGBRNI00P54D612B	2019
295	E99	Teorie dei processi comunic...	12	30	MSCCHR00P63D6...	2019

Si applica la funzione per eliminazione dell'esame:

```
SELECT(eliminaesamesuperato('E99'));
```

Data Output | Explain | Messages | Notifications

	codesame [PK] character varying (4)	nome character varying (70)	cfu smallint	voto smallint	studente character (16)	anno smallint
280	E85	Istologia C.I.	8	25	PCHCLT00H52A56...	2019
281	E86	Anatomia, istologia e fisiolo...	6	27	DCLRRT00R19D61...	2019
282	E87	Informatica	3	24	PCHCLT00H52A56...	2019
283	E88	Chimica analitica	12	26	FBBGDI00D61A851X	2019
284	E89	Chimica generale e inorganica	6	19	RSSSMN00S30D61...	2019
285	E9	TEORIE DEI PROCESSI COM...	12	25	MSCCHR00P63D6...	2019
286	E90	Antropologia di genere	12	27	MSCCHR00P63D6...	2019
287	E91	Laboratorio greco 2	3	29	SMLLSS00M67E20...	2019
288	E92	Linguistica generale	12	25	RLNDNS00T19F03...	2019
289	E93	Fisica	5	22	PCHCLT00H52A56...	2019
290	E94	Statistica I	9	20	CLMGLI01B54D612X	2019
291	E95	Scienze umane	4	19	PCHCLT00H52A56...	2019
292	E96	Psicologia Sociale	9	18	PLNTMS00R20I726P	2019
293	E97	Microbiologia	1	21	SGRNCHR00B68A8...	2019
294	E98	Chimica Generale e Inorgani...	9	21	SGBRNI00P54D612B	2019

Eliminando l'esame per la ridondanza devono essere modificati i valori di Media e CFUTot relativi allo studente il cui esame è stato eliminato:

Prima:

Data Output | Explain | Messages | Notifications

	studente [PK] character (16)	anno [PK] smallint	isee numeric (8,2)	cfutot smallint	media numeric (4,2)	attivo character (2)	pfi integer
93	MRCVCN01T05F839Z	2020	6764.46	6	30.00	SI	316
94	MRGLNE00R67A564F	2019	11998.79	18	22.00	SI	89
95	MRGLNE00R67A564F	2020	12465.79	0	0.00	SI	319
96	MRLRNI00A46A564E	2019	12834.69	0	0.00	SI	90
97	MRLRNI00A46A564E	2020	12860.69	0	0.00	SI	320
98	MRTGDI00E44F205G	2019	23006.30	4	29.00	SI	93
99	MRTGDI00E44F205G	2020	24713.30	0	0.00	SI	323
100	MSCCHR00P63D612N	2019	15669.67	36	27.33	SI	96
101	MSCCHR00P63D612N	2020	17906.67	0	0.00	SI	327
102	MSNLRI01R58D612S	2020	26278.94	9	23.00	SI	328
103	NLLLSS99E56A390O	2019	4129.00	24	26.63	SI	100
104	NLLLSS99E56A390O	2020	3966.00	0	0.00	SI	333
105	NNNLSS00L42B036T	2019	12551.34	0	0.00	NO	101
106	NTLGCM00S26A564F	2019	18005.85	24	21.25	SI	102
107	NTLGCM00S26A564F	2020	19024.85	0	0.00	SI	336
108	NTNLSI01C55A390I	2020	12493.54	9	30.00	SI	337
109	PCHCLT00H52A564K	2019	17309.61	35	20.97	SI	103
110	PCHCLT00H52A564K	2020	18192.61	12	28.33	SI	340

Dopo:

Data Output | Explain | Messages | Notifications

	studente [PK] character (16)	anno [PK] smallint	isee numeric (8,2)	cfutot smallint	media numeric (4,2)	attivo character (2)	pfi integer
91	MRCMTT00L24A564Y	2019	22501.52	42	24.29	SI	86
92	MRCMTT00L24A564Y	2020	23672.52	0	0.00	SI	314
93	MRCVCN01T05F839Z	2020	6764.46	6	30.00	SI	316
94	MRGLNE00R67A564F	2019	11998.79	18	22.00	SI	89
95	MRGLNE00R67A564F	2020	12465.79	0	0.00	SI	319
96	MRLRNI00A46A564E	2019	12834.69	0	0.00	SI	90
97	MRLRNI00A46A564E	2020	12860.69	0	0.00	SI	320
98	MRTGDI00E44F205G	2019	23006.30	4	29.00	SI	93
99	MRTGDI00E44F205G	2020	24713.30	0	0.00	SI	323
100	MSCCHR00P63D612N	2019	15669.67	24	26.00	SI	96
101	MSCCHR00P63D612N	2020	17906.67	0	0.00	SI	327
102	MSNLRI01R58D612S	2020	26278.94	9	23.00	SI	328
103	NLLLSS99E56A390O	2019	4129.00	24	26.63	SI	100
104	NLLLSS99E56A390O	2020	3966.00	0	0.00	SI	333
105	NNNLSS00L42B036T	2019	12551.34	0	0.00	NO	101
106	NTLGCM00S26A564F	2019	18005.85	24	21.25	SI	102
107	NTLGCM00S26A564F	2020	19024.85	0	0.00	SI	336

OP11: Stampa gli studenti universitari che rispettano gli obiettivi di merito in un dato anno

Si richiama la funzione per stampare gli studenti universitari che rispettano gli obiettivi di merito in un anno specifico:

```
SELECT
("stampa_universitari_anno_rispettano_merito"('2019')
);
```

	stampa_universitari_anno_rispettano_merito record
1	(MSCCHR00P63D612N,27.33,36)
2	(RCHMTT00E20D612F,29.00,24)
3	(RGGLRIOOS42D612K,27.14,21)
4	(SCHGLI00E44I046J,28.50,18)

OP21: Stampa il numero di CFU ottenuto da ogni studente universitario in un dato anno

Si richiama la funzione per stampare il numero di CFU ottenuto da ogni studente in un anno specifico:

```
SELECT ("CFU_Studente_Anno"('2019'));
```

	CFU_Studente_Anno record
1	(BLTGRL00M15A390W,24)
2	(BNCLRA00R59Z112H,0)
3	(BRGDSR00T61A564G,0)
4	(BRGMTT00M12A390I,51)
5	(BRGSLV00M55B036C,8)
6	(BRRLVO00A46D612U,7)
7	(BRRMRO01A09D612J,45)
8	(BRZMNL00E05A564N,6)
9	(BVIFNC00S19D612U,12)
10	(CHRGLI00D50D612I,33)
11	(CLBGNN00T29D612O,36)

OP23: Modifica dei CFU ottenuti da uno studente universitario in un dato esame

Si vuole modificare i CFU relativi ad un esame superato da uno studente universitario.
Si parte da questo esame e si vuole modificare il numero di CFU da 6 a 9:

	codesame [PK] character varying (4)	nome character varying (70)	cfu smallint	voto smallint	studente character (16)	anno smallint
1	E1	Statistica Computazionale	6	20	CLMGLI01B54D612X	2019

Si applica poi la funzione per modificare il numero di CFU dell'esame:

```
SELECT ("Modifica_Esame_CFU"('E1', '9'));
```

Si ottiene dunque:

	codesame [PK] character varying (4)	nome character varying (70)	cfu smallint	voto smallint	studente character (16)	anno smallint
1	E1	Statistica Computazionale	9	20	CLMGLI01B54D612X	2019

Modificando il numero di CFU per la ridondanza deve essere modificato il valore CFUTot relativo allo studente il cui esame è stato modificato:

Prima:

	studente [PK] character (16)	anno [PK] smallint	isee numeric (8,2)	cfutot smallint	media numeric (4,2)	attivo character (2)	pfi integer
1	CLMGLI01B54D612X	2019	15918.00	42	22.71	SI	28

Dopo:

	studente [PK] character (16)	anno [PK] smallint	isee numeric (8,2)	cfutot smallint	media numeric (4,2)	attivo character (2)	pfi integer
1	CLMGLI01B54D612X	2019	15918.00	45	22.00	SI	28

OP29: Aggiunta di una spesa

Supponendo di partire dalla tabella mostrata di “Spese” si applica la funzione per l’aggiunta di un nuova Spese.

Data Output

	codspesa [PK] integer	quantita numeric (8,2)	areaspese character varying (3)	pfi integer
1332	1333	1087.70	S8	131
1333	1334	1548.60	S2	132
1334	1335	1185.80	S7	133
1335	1336	831.00	S10	134
1336	1337	436.50	S2	135
1337	1338	1350.90	S6	136
1338	1339	838.00	S8	137
1339	1340	1201.60	S5	138
1340	1341	1203.60	S7	139
1341	1342	1097.20	S9	140
1342	1343	1305.70	S12	141
1343	1344	656.10	S6	142
1344	1345	1232.80	S7	143
1345	1346	1626.00	S2	144
1346	1347	828.10	S7	145
1347	1348	976.10	S8	146
1348	1349	742.30	S5	147
1349	1350	953.40	S7	148
1350	1351	933.10	S10	149
1351	1352	1103.90	S10	150
1352	1353	1299.30	S8	151
1353	1354	683.90	S8	152
1354	1355	579.80	S9	153
1355	1356	2220.10	S2	154
1356	1357	1859.70	S10	155

Successivamente si inserisce una spesa usando la relativa funzione:

```
SELECT ("Aggiungi_Spesa"('1000','S10','155'));
```

Data Output

	codspesa [PK] integer	quantita numeric (8,2)	areaspese character varying (3)	pfi integer
1333	1334	1548.60	S2	132
1334	1335	1185.80	S7	133
1335	1336	831.00	S10	134
1336	1337	436.50	S2	135
1337	1338	1350.90	S6	136
1338	1339	838.00	S8	137
1339	1340	1201.60	S5	138
1340	1341	1203.60	S7	139
1341	1342	1097.20	S9	140
1342	1343	1305.70	S12	141
1343	1344	656.10	S6	142
1344	1345	1232.80	S7	143
1345	1346	1626.00	S2	144
1346	1347	828.10	S7	145
1347	1348	976.10	S8	146
1348	1349	742.30	S5	147
1349	1350	953.40	S7	148
1350	1351	933.10	S10	149
1351	1352	1103.90	S10	150
1352	1353	1299.30	S8	151
1353	1354	683.90	S8	152
1354	1355	579.80	S9	153
1355	1356	2220.10	S2	154
1356	1357	1859.70	S10	155
1357	1358	1000.00	S10	155

Aggiungendo una spesa per la ridondanza viene modificato il valore di SpeseTot nella tabella PFI relativamente al PFI cui è stata aggiunta la spesa.

Prima:

	codpfi [PK] integer	minutidasvolgere smallint	quantitaborsa numeric (8,2)	spesetot numeric (8,2)	codtutor integer	categoria character (1)	minutitot smallint
1	155	1100	3500.00	4483.70	3	G	195

Dopo:

	codpfi [PK] integer	minutidasvolgere smallint	quantitaborsa numeric (8,2)	spesetot numeric (8,2)	codtutor integer	categoria character (1)	minutitot smallint
1	155	1100	3500.00	5483.70	3	G	195